НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Інформаційно-телекомунікаційних мереж

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Лариса ГЛОБА
«___»_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інформаційно-комунікаційні технології»
зі спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Удосконалений метод обробки потоків в системі SDN**»**

Виконав:
студент VI курсу, групи ТІ-91мп
Шевченко Максим Віталійович _____

Керівник:
Професор кафедри ІТМ ІТС, д.т.н., с.н.с.
Скулиш Марія Анатоліївна _____

Рецензент:
Професор кафедри ТК ІТС, професор, д.т.н.
Лисенко Олександр Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Лариса ГЛОБА

«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Шевченку Максиму Віталійовичу

1. Тема дисертації «Удосконалений метод обробки потоків в системі SDN», науковий керівник дисертації професор кафедри інформаційно-телекомунікаційних мереж ІТС Скулиш Марія Анатоліївна, д.т.н., с.н.с., затверджені наказом по університету від «03» листопада 2020 р. № 3208-с.

2. Термін подання студентом дисертації 10.12.2020 р.

3. Об’єкт дослідження телекомунікаційна складова програмно-конфігуруються мереж, що включає канали зв'язку, контролер ПКС та мережеве обладнання.

4. Предмет дослідження методи керування потоками в контролері SDN.

5. Перелік завдань, які потрібно розробити:

1) Аналіз SDN мереж, особливості передачі в SDN.
2) Аналіз існуючих рішень для розробки додатків в SDN контролері.
3) Аналіз середовища Mininet.
4) Моделювання мережі за допомогою середовища Mininet в VM.
5) Встановлення OpenDayLight контролер в VB.

6) Інтеграція OpendayLight в Mininet.

6. Орієнтовний перелік ілюстративного матеріалу

1. Тема, актуальність, мета, задачі.
2. Особливості передачі в SDN
3. Контролери для розробки SDN додатку.
4. Середовище моделювання мережі.
5. Розгортання мережі.
6. Встановлення контролеру.
7. Загальні висновки.

7. Дата видачі завдання 03.09.2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Аналіз SDN мереж, особливості передачі в SDN	03.09.2020 – 05.10.2020	виконано
2.	Аналіз існуючих рішень для розробки додатків в SDN контролері	05.10.2020-21.10.2020	виконано
3.	Аналіз середовища Mininet	21.10.2020-28.10.2020	виконано
4.	Моделювання мережі за допомогою середовища Mininet в VM	28.10.2020-4.11.2020	виконано
5.	Встановлення OpenDayLight контролер в VB	4.11.2020-16.11.2020	виконано
6.	Інтеграція OpendayLight в Mininet	16.11.2020-30.11.2020	виконано
7.	Розробка стартап-проекту	30.11.2020 – 3.12.2020	виконано
8.	Підготування графічного матеріалу	4.12.2020 – 7.12.2020	виконано

Студент Максим ШЕВЧЕНКО

Науковий керівник дисертації Марія СКУЛИШ

РЕФЕРАТ

Робота містить 81 сторінок, 28 рисунків та 4 таблиці. Було використано 37 джерела.

Мета роботи: поліпшити ефективність алгоритму обрання внутрішнього шляху в системі.

Проведено огляд технології програмно-конфігурованих мереж, описана архітектура SDN, описані OpenFlow протокол, конвеєрна обробка пакетів, архітектура Open vSwitch комутатор, основні функції та вимоги до OpenFlow контроллеру. Висвітлені особливості контролера SDN, переваги та недоліки найбільш популярних зовнішніх контролерів таких, як: OpenDayLight, POX, Floodlight. Розглянуто емулятор Mininet та практично описані функції Mininet, і розгорнута власна топологія. Зроблений опис розгортання обраного контролеру OpenDayLight на віртуальну машину в Oracle VM VirtualBox.

До недоліків можна віднести практична частина не доведена до кінця так, як виникла проблема з підключенням зовнішнього контролеру до емулятору, але описана вище робота дає змогу поставити остаточну оцінку роботи.

Ключові слова: SDN. OpenDayLight, SmartCity, POX, Floodlight, Mininet

ABSTRACT

The work contains 81 pages, 28 figures and 4 table. 37 sources have been used.

Goal: improve the efficiency of the algorithm for selecting the internal path in the system

An overview of software-configured network technologies is described, SDN architecture is described, OpenFlow protocol is described, packet pipeline processing, Open vSwitch switch architecture is described, main functions and requirements for OpenFlow controller are described. Features of the SDN controller, advantages and disadvantages of the most popular external controllers such as: OpenDayLight, POX, Floodlight are covered. The Mininet emulator is considered and the Mininet functions are practically described, and the own topology is developed. The description of deployment of the selected OpenDayLight controller on the virtual machine in Oracle VM VirtualBox is made.

The disadvantages include the practical part is not completed as there was a problem with the connection of the external controller to the emulator, but the work described above allows you to make a final assessment of the work.

Keywords: SDN, OpenDayLight, SmartCity, POX, Floodlight, Mininet

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ACL	Access Control List
API	Application Programming Interface
BGP	Border Gateway Protocol
CLI	Command line interface
ECMP	Equal-cost multi-path routing
GNU	GNU's Not UNIX
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IS-IS	Intermediate System-to-Intermediate System
JSON	JavaScript Object Notation
LACP	Link Aggregation Control Protocol
MAC	Media Access Control
NBI	NorthBound
NETCONF	Network Configuration Protocol
ONF	Open Networking Foundation
OSGI	Open Services Gateway Initiative
OSPF	Open Shortest Path First
OVS	Open vSwitch
OVSDB	Open vSwitch Database
OXM	OpenFlow Extensible Match fields
PCEP	Path Control Element Protocol
PDU	Power Distribution Unit
QoS	Quality of Service
REST	Representational State Transfer
SDN	Software-defined Networking
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
TLS	Transport layer security
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
XML	eXtensible Markup Language
ОС	Операційна система
ПКС	Програмно конфігуровані мережі
ПЗ	Програмне забезпечення

ВСТУП

Актуальність теми: Сучасний стан і тенденції розвитку комп'ютерних мереж показали, що потенціал зростання продуктивності, пропускної спроможності мереж на основі традиційних технологій практично вичерпаний. Це пов'язано із зростанням витрат часу на маршрутизацію, з труднощами налаштування мережі і управління потоками в ній. Особливо з урахуванням нових потреб в якості сервісів для високошвидкісних глобальних мереж і мереж центрів обробки даних. Також з ростом потреб у віртуалізації мереж, тобто відображення декількох логічно ізольованих мереж з незалежними політиками якості обслуговування на загальний набір мережевих ресурсів.

Таке незадовільний стан справ може змінитися через двох революційних подій: перше, поява на ринку надзвичайно ускладненого, пропрієтарного, мережевого обладнання, і друге, поява принципово нового підходу, званого программноконфігуріруемимі мережами (ПКС - Software Defined Networks). Дотримання такого підходу дозволить прискорити маршрутизацію в мережах, зробити його зручнішим для конфігурації, віртуалізації, настройки якості обслуговування, але вимагає додаткових досліджень і розробок. Зокрема, в області організації мережевих комутаторів, програмних додатків для управління мережею і платформ для їх виконання.

Об’єкт дослідження: телекомунікаційна складова програмно-конфігуруються мереж, що включає канали зв'язку, контролер ПКС та мережеве обладнання.

Предмет дослідження: методи керування потоками в контролері SDN.

Мета роботи. підвищення швидкодії роботи програмно-керованої мережі за рахунок вдосконалення процесу керування потоками.

Теоретичний реультат досліження:

Досліджено використання середовища Mininet та контролерів SDN для розгортування мережі SDN, що дозволяє виявити та усунути проблеми мережі на етапі моделювання.

Практичний результат дослідження.

Математична модель додатку SDN в майбутньому що дозволить зменшити час на розробку подібних SDN додатків, реаліціонна модель для веріфікіцаії

РОЗДІЛ 1

SDN МЕРЕЖА, ОСОБЛИВОСТІ І ПЕРЕДАЧА В SDN

1.1 Архітектура SDN

Архітектура SDN ґрунтується на логічному і фізичному поділі функцій рівнів мережевих пристроїв - відділенні Control Plane від Data Plane і перенесення площини управління мережею на виділений SDN контролер, який має точну інформацію про структуру і топології мережі. Таким чином, в рамках класичної концепції SDN мережевий пристрій - передавальний пристрій без функцій управління [1].

SDN складається з трьох рівнів:

- рівень інфраструктури;
- рівень управління;
- рівень додатків.

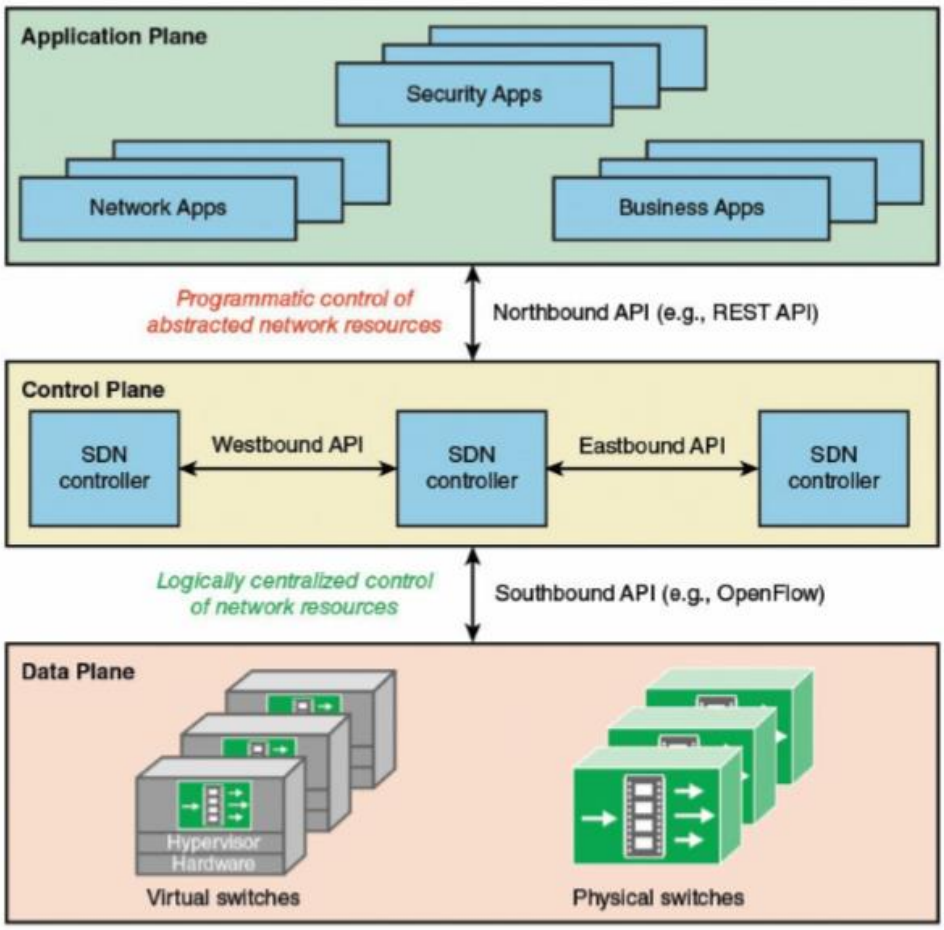

Рис 1.1 Архітектура Software-defined network

Опис рівнів SDN:

а) рівень інфраструктури або передачі даних (data plane) - включає в себе набір передавальних мережевих пристроїв, які доступні через уніфікований інтерфейс і виконують інструкції таблиць потоків для передачі мережевого трафіку. Це можуть бути як фізичні, так і віртуальні комутатори, і маршрутизатори, а також канали зв'язку [2-3];

б) рівень управління (control plane) «контрольна площина» - регулює обмін інформацією про таблиці маршрутизації і виробляє правила маршрутизації на основі закладених в контролер алгоритмів аналізу потоків пакетів, що пересилаються від комутаторів. Реалізується на самому контролері SDN або кластері таких контролерів, що забезпечують функції управління мережевою інфраструктурою.

На даному рівні є три інтерфейсу для взаємодії з іншими елементами мережевої інфраструктури. За допомогою цих інтерфейсів рівень управління пов'язує всі три площини [1-3]:

- Southbound API - (спадний інтерфейс для рівня передачі даних) забезпечує взаємодію SDN контролера з мережевими пристроями за допомогою спеціальних протоколів, наприклад, OpenFlow. Це дозволяє контролеру динамічно вносити зміни і конфігурувати мережу в реальному часі. Інтерфейс здатний підтримувати декілька протоколів (у вигляді окремих модулів), таких як OpenFlow 1.0, OpenFlow 1.3 і BGPLS;

- Northbound API - (для рівня додатків) інтерпретує бізнес логіку в мережеві інструкції, надає основні мережеві функції, такі як обчислення шляху даних, маршрутизація і безпеку, дозволяє гнучко виділяти мережеві ресурси, виходячи з вимог додатків, абстрагуючись мережеву інфраструктуру. Використовується для автоматизації та управління даними. Даний інтерфейс ще не стандартизований . Крім того, більшість контролерів в якості інтерфейсу висхідного взаємодії підтримують RESTful API. API-інтерфейси Northbound можуть використовувати Python, Java, C, REST, XML, JSON.

Інтерфейс NorthBound (NBI) можна назвати межею між контролером і прикладним рівнем. Він підтримує більшість протоколів рівня додатки для взаємодії, тобто HTTPS, SFTP і т. д. Ми можемо керувати контролером за допомогою основних HTTP методів POST, GET, PUT і DELETE;

- східний / західний інтерфейс (для взаємодії між елементами рівня управління). Відстежує топологію всієї мережі і надає програмний інтерфейс (API) для мережевих додатків;

в) рівень додатків складається з високорівневих додатків, які взаємодіють з мережевим контролером або набором контролерів, запитуючи і передаючи необхідні ресурси та інструкції за допомогою API для реалізації конкретних функцій, що відповідають вимогам мережевої інфраструктури. Ця взаємодія здійснюється за допомогою такого компонента SDN як Northbound API. Тут реалізуються різні функції обробки мережевого трафіку. Прикладом таких додатків можуть служити засоби моніторингу (збір інформації про мережу та відправка її SDN додатків), аналітики балансування навантаження або бізнес-додатки [2].

Southbound і Northbound інтерфейси представлені на рисунку 1.2.

Відділення управління від функцій форвардинга і розгляд кожного рівня як самостійної одиниці надає додаткам більш детальну інформацію від контролера про стан мережі, в порівнянні з традиційними мережами.

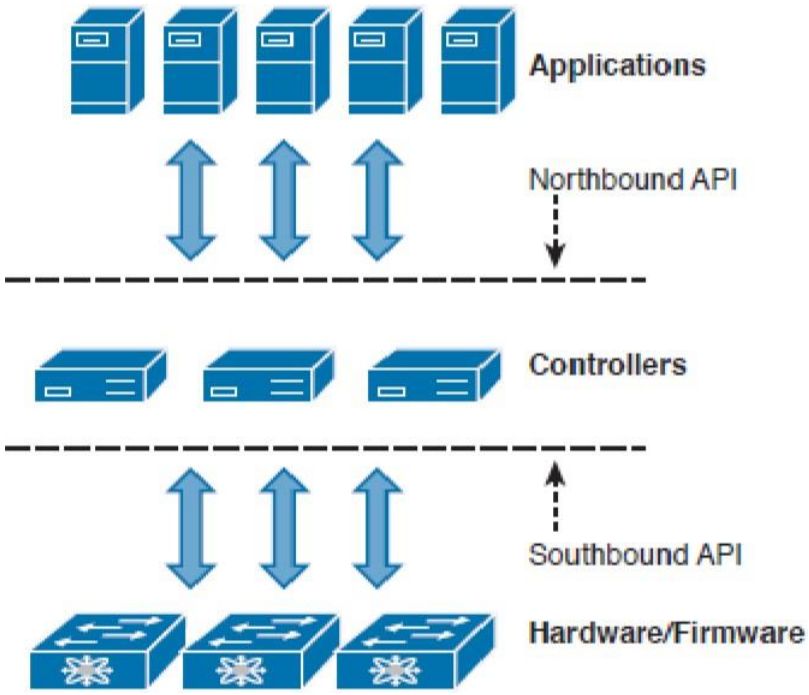

Рис. 1.2 Southbound і Northbound інтерфейси

1.2 OpenFlow Протокол

Найбільш перспективним і активно розвиваються стандартом для SDN є OpenFlow (OpenFlow версія 1.3) - відкритий стандарт, в якому описуються вимоги, що пред'являються до комутатора, що підтримує протокол OpenFlow для віддаленого управління [4].

За допомогою сучасних маршрутизаторів зазвичай вирішуються дві основні задачі: передача даних (forwarding) - просування пакета від вхідного порту на певний вихідний порт і управління даними - обробка пакету і прийняття рішення про те, куди його передавати далі, на основі поточного стану маршрутизатора [4] .

Це відповідає рівню передачі даних, на якому зібрані кошти передачі (лінії зв'язку, каналоутворювального обладнання, маршрутизатори, комутатори), і рівнем управління станами засобів передачі даних (рисунок 1.3). Розвиток маршрутизаторів досі йшло по шляху зближення цих рівнів, проте з ухилом на передачу (апаратне прискорення, вдосконалення ПО і впровадження нових функціональних можливостей для збільшення

швидкості прийняття рішення по маршрутизації кожного пакета), тоді як рівень управління залишався досить примітивним і спирався на складні розподілені алгоритми маршрутизації і хитромудрі інструкції по конфігурації і налаштування мережі. Зрозуміло, ПЗ маршрутизаторів, що реалізує рівень управління, було пропрієтарним і закритим [4].

Згідно зі специфікацією 1.3 стандарту OpenFlow, взаємодія контролера з комутатором здійснюється за допомогою протоколу OpenFlow - кожен комутатор повинен містити одну або більше таблиць потоків (flow tables), групову таблицю (group table) і підтримувати канал (OpenFlow channel) для зв'язку з віддаленим контролером - сервером [4].

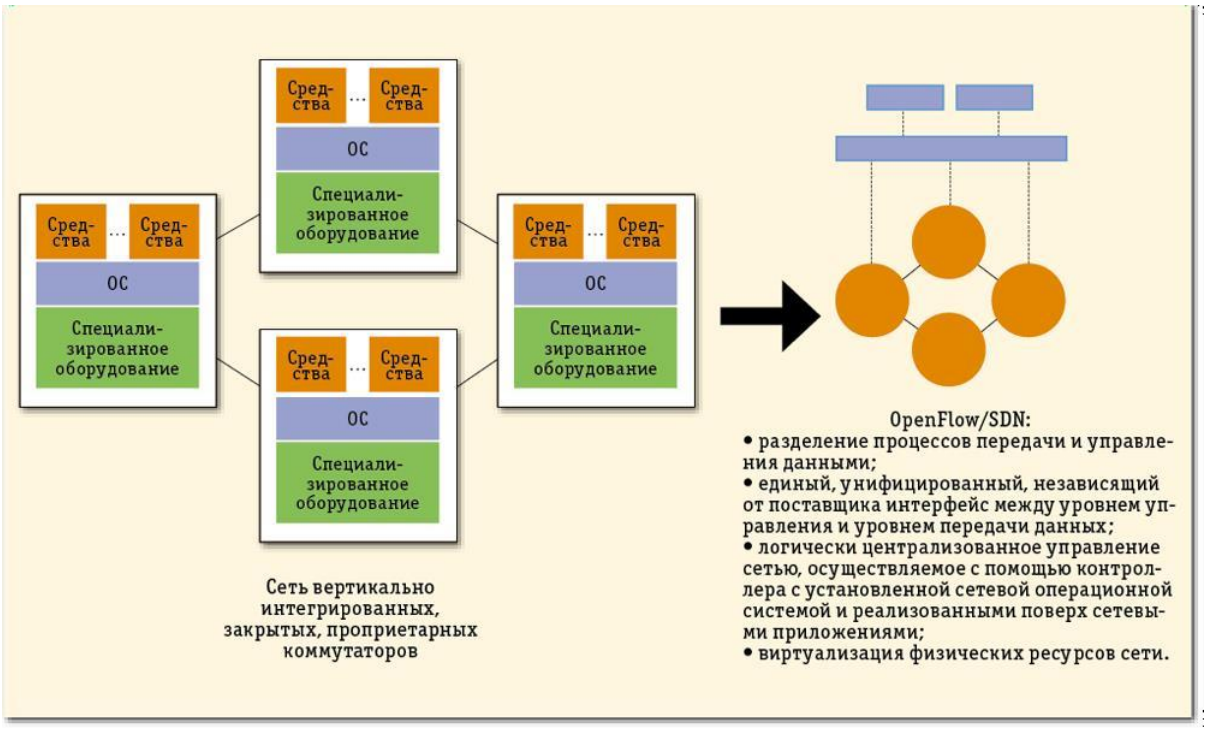

Рис. 1.3 Традиційні мережі і SDN

Специфікація не регламентує архітектуру контролера і API для його додатків. Кожна таблиця потоків в комутаторі містить набір записів (flow entries) про потоки або правила. Кожна такий запис складається з полей-ознак (match fields), лічильників (counters) і набору інструкцій (instructions) [4].

Механізм роботи комутатора OpenFlow досить простий. У кожного пакету, що прийшов «вирізується» заголовок (бітова рядок певної довжини).

Для цієї бітової рядки в таблицях потоків, починаючи з першої, шукається правило, у якого поле ознак найближче відповідає (збігається) заголовку пакета. При наявності збігу, над пакетом і його заголовком виконуються перетворення, які визначаються набором інструкцій, зазначених в знайденому правилі. Інструкції, асоційовані з кожним записом таблиці, описують дії, пов'язані з пересилкою пакета, модифікацією його заголовка, обробкою в таблиці груп, обробкою в конвеєрі і пересиланням пакета на певний порт комутатора. Інструкції конвеєра обробки дозволяють пересилати пакети в наступні таблиці для подальшої обробки і у вигляді метаданих передавати інформацію між таблицями. Інструкції також визначають правила модифікації лічильників, які можуть бути використані для збору різноманітної статистики [4].

Якщо потрібного правила в першій таблиці не виявлено, то пакет инкапсулируется і відправляється контролеру, який формує відповідне правило для пакетів даного типу і встановлює його на комутаторі (або на наборі керованих їм комутаторів), або пакет може бути скинутий (в залежності від конфігурації комутатора) [4].

Запис про потік може приписувати переслати пакет в певний порт (звичайний фізичний порт або віртуальний, призначений комутатором, або зарезервований віртуальний порт, встановлений специфікацією протоколу). Зарезервовані віртуальні порти можуть визначати загальні дії пересилання: відправка контролера, широковещательная (лавинна) розсилка, пересилання без OpenFlow. Віртуальні порти, певні комутатором, можуть точно визначати групи агрегування каналів, тунелі або інтерфейси зі зворотним зв'язком [4].

Записи про потоках можуть також вказувати на групи, в яких визначається додаткова обробка. Групи є наборами дій для широкомовної розсилки, а також набори дій пересилання з більш складною семантикою, наприклад швидка зміна маршруту або агрегування каналів. Механізм груп дозволяє ефективно змінювати загальні вихідні дії для потоків. Таблиця груп містить записи про групи, що містять список контейнерів дій зі спеціальною

семантикою, що залежить від типу групи. Дії в одному або декількох контейнерах дій застосовуються до пакетів, що відправляються в групу [4].

Розробники комутаторів можуть бути вільні у реалізації їх внутрішньої начинки, однак процедура перегляду пакетів і семантика інструкцій повинні бути для всіх однакові. Наприклад, в той час як потік може використовувати всі групи для пересилки в деякий безліч портів, розробник комутатора може вибрати для реалізації цього єдину бітову маску всередині апаратної таблиці маршрутизації. Інший приклад - це процедура перегляду таблиць: конвеєр фізично може бути реалізований за допомогою різної кількості апаратних таблиць. Установка, оновлення та видалення правил в таблицях потоків комутатора здійснюються контролером. Правила можуть встановлюватися реактивно (у відповідь на що прийшли пакети) або проактивно (заздалегідь, до приходу пакетів) [4].

Управління даними в OpenFlow здійснюється не на рівні окремих пакетів, а на рівні їх потоків. Правило в комутаторі OpenFlow встановлюється за участю контролера тільки для першого пакету, а потім всі інші пакети потоку його використовують [4].

1.3 OpenFlow комутатор

OpenFlow комутатор - мережевий пристрій Data Plane, що підтримує роботу по протоколу OpenFlow і виконує функції перенаправлення даних згідно з інструкціями SDN контролера. Завдання - передача та обробка пакетів відповідно до таблиці мережевих потоків і правил, сформованої контролером на підставі певних критеріїв кадру або пакета. Пристрої позбавлені функції управління і можуть тільки передавати статистику і звертатися за новими правилами потоків до зовнішнього SDN-контролера [3].

Згідно зі специфікацією стандарту, OpenFlow - кожен комутатор повинен містити одну або більше таблиць потоків (flow tables), групову таблицю (group table) і підтримувати канал (OpenFlow channel) для зв'язку з

віддаленим контролером по протоколу OpenFlow [1]. Компоненти OpenFlow комутатора представлені на рисунку 1.4 [3], [5-6].

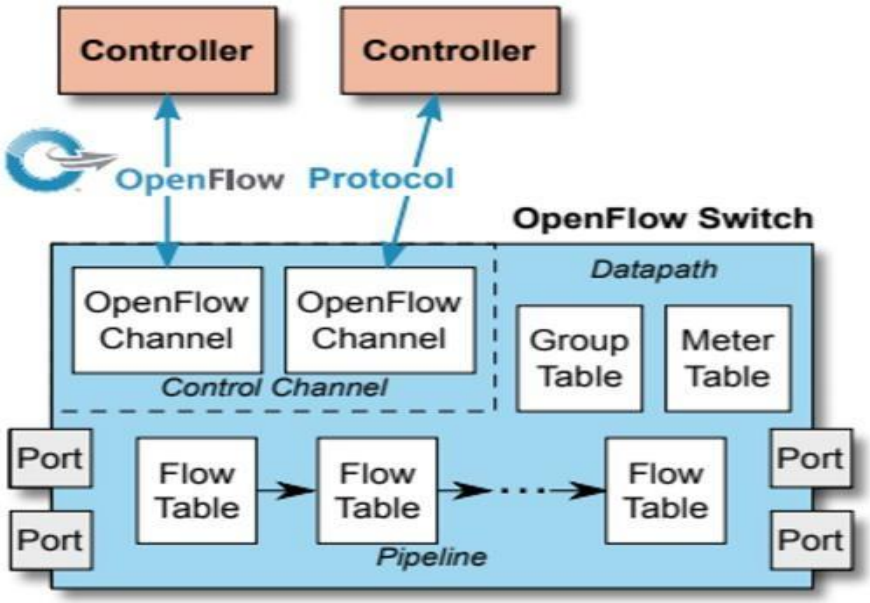

Рис. 1.4 Компоненти OpenFlow комутатора

Канал OpenFlow (OpenFlow Channel) - інтерфейс, який з'єднує кожен логічний комутатор OpenFlow з контролером. Для кожного контролера, що підключається до комутатора, встановлюється окремий канал [5-6].

Виділяють два підходи до побудови такого каналу:

а) Out-of-band - передача повідомлень протоколу по фізично незалежного каналу (використовується окремий порт комутатора), не використовуючи мережу передачі даних. Забезпечує більш високу надійність і безпеку за рахунок відсутності впливу переданого по мережі трафіку на керуючий трафік і виключення фізичного доступу до мережі управління з боку вузлів в мережі передачі даних;

б) In-band - передача керуючої інформації через мережу передачі даних. Менш надійний і безпечний спосіб. До переваг підходу можна віднести більш високу економічність і скорочення кількості мережевого обладнання, спрощення проектування і мережі.

Формат записів в таблиці мережевих протоколів OpenFlow комутатора представлений на рисунку 1.5.

Формат заголовка Flow table представлений на рисунку 1.6.

OpenFlow порт - мережевий інтерфейс для передачі пакетів між пристроями з підтримкою OpenFlow. Виконують функції портів стандартного L2-комутатора. Пакети, що приходять на OpenFlow-порт, класифікуються за потокам в таблицях потоків за допомогою Match класифікаторів. Детальна структура таблиці потоків представлена на рисунку 1.8.

Сравнение(match fields	Приоритет (priority)	Счетчики(counters)	Действия (Actions)	Тайм-ауты(time-outs)	Метаданные

Рис. 1.5 Формат записів в таблиці мережевих протоколів OpenFlow комутатора

Ingress port	MAC Src	MAC dstn	Eth type	Vlan ID	Vlan Priority	IP src	IP dstn	IP Pport	TCP sPort	TCP dPort	Action

Рис. 1.6 Формат заголовка Flow table

Обробка пакетів на основі таблиці представлена на рисунку 1.7.

Записи в таблиці мережевих протоколів OpenFlow комутатора:

- match - правило виділення пакетів, що належать конкретній потоку за допомогою порівняння заголовків пакету (дані L2-L4). Набір полів «match» заданий в спеціальному форматі OpenFlow Extensible Match fields (OXM);

- actions - визначає яким чином будуть оброблені елементи даного потоку. У специфікації OpenFlow і Open vSwitch вказані основні дії, але також є можливість розширення [1], [3], [5-6];

- статистика - відстежує кількість прийнятих пакетів конкретного потоку і кількість байт за допомогою поновлення фрейм-лічильників. Лічильники доступні для таблиць, потоків, портів і черг;

- priority - числове поле від 0 до 65535;

- timeouts - максимальний час до видалення відповідної flow записи;

- метадані - не використовуються при обробці пакетів. Може використовуватися контролером для фільтрації статистики потоку, зміни потоку і видалення потоку, використовується для перенесення інформації від однієї таблиці до іншої.

Відповідний запис визначається по полях порівняння (поле «protocol») і пріоритету. При збігу полів вибирається запис з найбільшим пріоритетом. Відповідно до неї буде оброблятися пакет [3]. Процес обробки пакета представлений на малюнку.

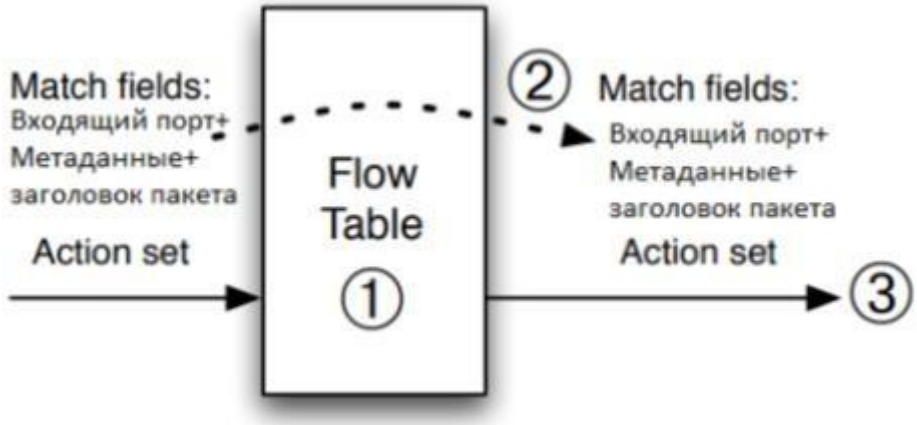

Рис. 1.7 Обробка пакетів на основі таблиці

Другим елементом записи таблиці є «дія» (action), яке зіставляється кожного запису в таблиці потоків, застосовується до пакетів, що належать даному потоку, і визначає обробку пакетів потоку [5-6]. Основними діями є: пересилання на певний порт або кілька портів, відкидання всіх пакетів конкретного потоку, перетворення пакета (модифікація заголовка пакета) - інкапсуляція і передача пакетів на контролер ПКС по безпечному каналу. Крім того, «стандартна» обробка, дозволяє розділити потоки даних на потоки, керовані OpenFlow, і потоки, керовані іншими механізмами (існуючі протоколи маршрутизації). Надає можливість ізолювати експериментальний трафік, використовуючи загальну інфраструктуру.

У OpenFlow є механізм обробки пакетів з використанням груп OpenFlow. Група - це набір підмножин, кожна підмножина складається з набору Actions, які можуть бути наборами дій для широкомовної розсилки і більш складних механізмів пересилання.

Функціонал GROUPS розширює можливості пересилання пакетів. Наприклад, дозволяє реалізувати ECMP Load Balancing - «відправка на одне підмножина з групи» - для балансування навантаження в LAG або ECMP.

Таблиця груп (group table) містить записи про групи, що містять список контейнерів дій зі спеціальною семантикою в залежності від типу групи.

Механізм роботи OpenFlow комутатора досить простий. Починаючи з версії 1.3 OpenFlow підтримує множинні таблиці потоків. Для обробки потоків, комутатором використовується механізм конвеєрної обробки (pipeline) - обробка пакету, що прийшов починається до закінчення обробки попереднього [3], [5-6].

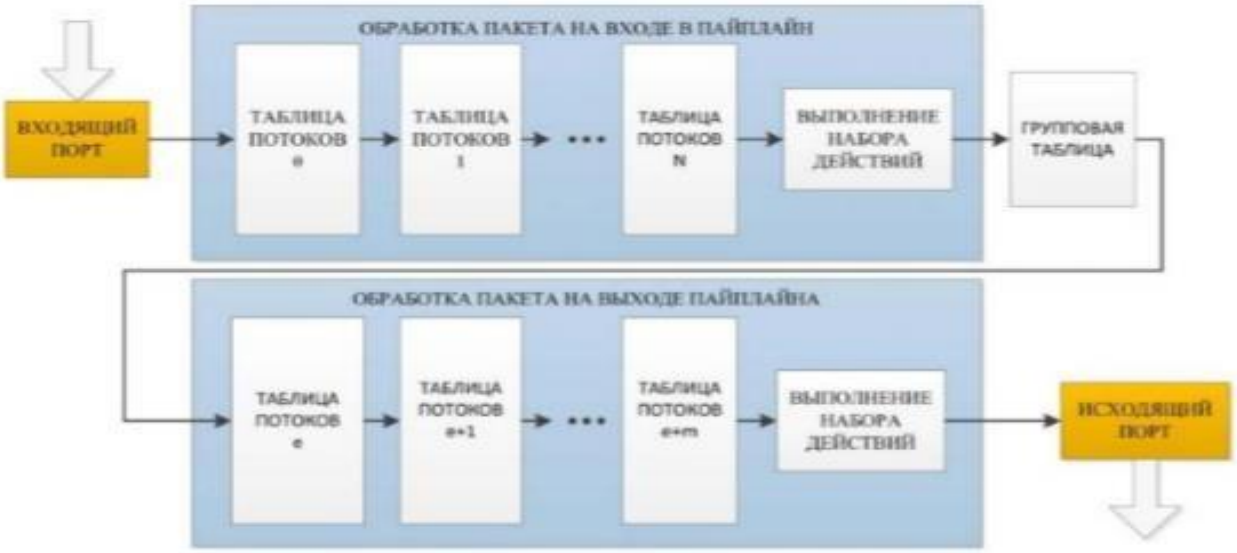

Рис. 1.8 Конвеєрна обробка пакета

Конвеєр (Pipeline) - набір пов'язаних таблиць, які забезпечують перевірку заголовків, пересилання і модифікацію пакетів в OpenFlow комутаторі.

OpenFlow агент отримує команди від контролера і формує таблицю flow, яка містить інструкції по обробці прийшов PDU пакета. Таким чином OpenFlow пов'язує пристрою управління і пристрої передачі.

OpenFlow широко впроваджується виробниками мережевого устаткування через простої структури OpenFlow-комутатора, яка може бути реалізована за рахунок невеликих модифікацій програмного і апаратного забезпечення. В результаті перехід на протокол OpenFlow може бути проведений покроково з впровадженням протоколу в ті мережеві сегменти, які вимагають функцій OpenFlow.

1.4 Open vSwitch комутатор

Віртуальний комутатор - ключовий компонент для віртуалізації мережевої інфраструктури. Він з'єднує віртуальні мережеві адаптери з фізичними мережевими адаптерами, встановленими на сервері, встановлює зв'язок між віртуальними мережевими адаптерами для локального взаємодії в рамках сервера.

На даний момент існує три найбільш популярних віртуальних комутатора - VMware virtual switch (standard і distributed), Cisco Nexus 1000v і Open vSwitch.

Open vSwitch - найбільш універсальне рішення. Багаторівневий віртуальний комутатор з відкритим вихідним кодом, що розробляється під ліцензією Apache 2.0. Повністю керований, незалежний комутатор, який реалізує віртуальну комутацію на базі протоколу OpenFlow. Підтримка OpenFlow надає можливість для інтеграції промислових хмарних систем з існуючою ПКС. Сумісний з популярними платформами для комутації за рахунок програмних додатків.

Підтримує найбільш популярні Гіпервізор, включаючи KVM, Virtual Box, Xen та XenServer. Open vSwitch складається зі служби-комутатора (user-space) і супроводжуючого модуля ядра (kernelspace), який керує процесом потокової (flowbased) комутації.

Open vSwitch підтримує широкий набір технологій, включаючи NetFlow, sFlow, port mirroring, VLAN, LACP, TLS / SSL.

В рамках класичної моделі SDN для управління OVS (формування FIB) застосовуються сторонні компоненти. наприклад, плагін для OpenStack Neutron або SDN-контролер OpenDayLight.

Також OVS можна використовувати в режимі стандартного комутатора, без зовнішнього керуючого елемента і застосовувати MAC learning для формування таблиць комутації.

1.4.1Структура Open vSwitch

Open vSwitch - це віртуальний багаторівневий (multilayer) маршрутизатор розробляється під ліцензією Apache 2.0.

Open vSwitch умовно можна розділити на дві частини - user-space і kernel-space.

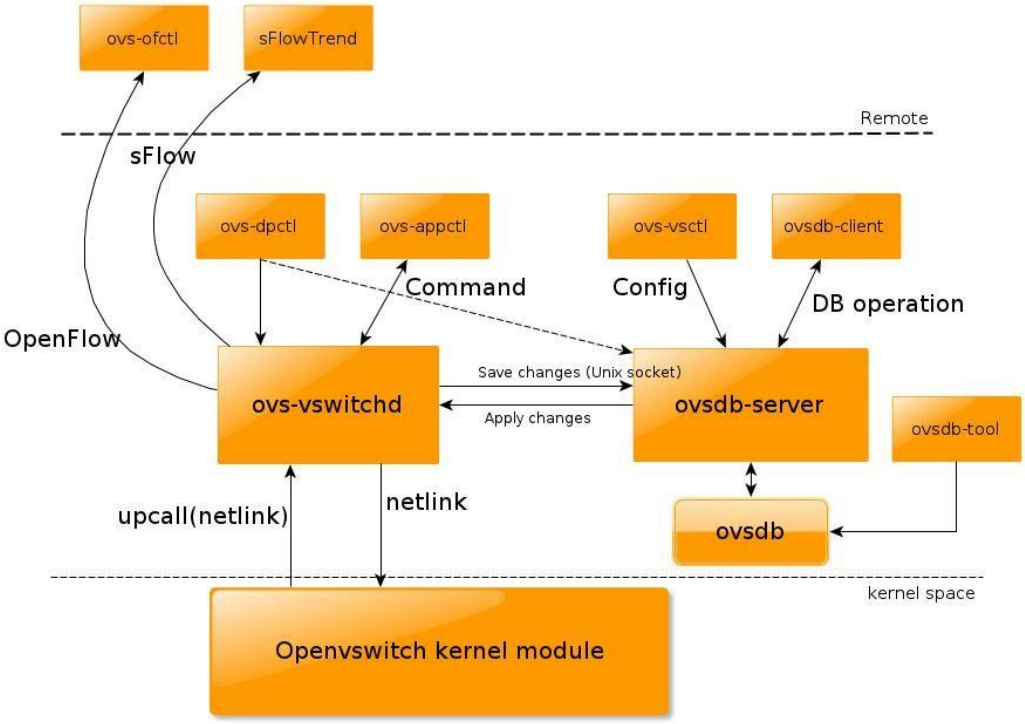

Рис. 1.9 Структура Open vSwitch (www.openvswitch.org)

User-space складається з декількох найбільш важливих компонентів: це демони (daemons), які реалізують свитч з таблицею потоків, ядро Open vSwitch, і набір утиліт, які дозволяють конфігурувати свитч, його базу даних (ovsdb) і безпосередньо посилати повідомлення в ядро.

Openvswitch service запускає три демона: ovs-vswitchd - зберігає і відповідає за зміну конфігурації маршрутизатора, а також зберігає стан в базу даних (ovsdb), ovsdb-server - маніпулює базою даних, конфігурацією і набором потоків, ovs-brcompatd - створює сумісність Open vSwitch зі звичайними Linux мостами (які створюються командою brctl).

Демон ovs-vswitchd - єдина компонента, пов'язана з kernel-space через протокол netlink, також змінює конфігурацію Open vSwitch і зберігає її в базу

даних (ovsdb), яка керується за допомогою демона ovsdb-server (комунікація ovs-vswitchd і ovsdb-server здійснюється за допомогою сокетів)

• ovs-dpctl - дозволяє безпосередньо звертатися до kernel-space, без сторонніх звернень до бази даних.

• ovsdb-client - утиліта для конфігурації бази даних ovsdb.

• ovs-ofctl - команда для роботи з протоколом OpenFlow, модифікацією flow- таблиць, налаштування з'єднання з віддаленим контролером.

Варто відзначити, що Open vSwitch є гібридним маршрутизатором - крім підтримки OpenFlow, маршрутизатор підтримує спеціалізовані команди (Nicira Extensions) (L2 routing, QoS, tunneling і.т.д), але найбільший інтерес він представляє як проект що підтримує протокол OpenFlow (на даний момент є підтримка OpenFlow 1.0 і тестові версії з підтримкою OpenFlow 1.3).

1.5 Контролер OpenFlow і його функції

SDN Контролер - керуючий центр мережі, що включає центральну мережеву операційну систему і керуючі програми. Відповідає за побудову і відображення топології, програмування мережевих пристроїв і служить єдиною точкою управління всією мережею. SDN контролер надає можливість об'єднувати кілька контролерів, і встановлювати ієрархічні зв'язки між ними [1-2]. SDN контролер реалізується в якості ПО для управління площиною пересилання.

Платформа контролера містить набір динамічно модулів для виконання необхідних мережевих завдань за допомогою програмування функцій мережевої інфраструктури. Це є фундаментальним властивістю архітектури SDN та її основною перевагою. Поділ функціональних особливостей додатків дозволяє здійснювати кілька завдань в мережі одночасно.

Таблиця 1.1

Основні функції контролера

Керування ресурсами сервера	поддержка многопоточности; мониторинг загрузки сервера.
Забезпечення зв'язку між рівнем додатків і мережею	- взаємодія додатку-комутатор; - динамічна конфігурація мережі в залежності від параметрів трафіку;
Надання сервісів	- побудова топології; - моніторинг хостів; - додавання / видалення нових елементів мережі.
Керування комутацією	- програмування flow table і правил пересилання трафіку.
Створення додатків на основі API	- одночасний запуск декількох додатків; - реєстрація додатків; - обробка помилок додатків і помилок взаємодії додатків; - підписка на події від додатків; - пріоритизація додатків; - забезпечення взаємодії між додатками за допомогою відповідної інфраструктури; - балансування навантаження додатків (по потокам); - розмежування прав доступу додатків до елементів мережі.

Відкритий програмний інтерфейс API (від англ. Application Programming Interface), дозволяє розробляти програми і програмні розширення, які реалізують логіку, необхідну, для визначення, поновлення та адаптації правил потоків.

Потік пакетів - послідовність пакетів, що проходять через мережу, с загальним набором значень полів заголовка, представлених на малюнку 6.

Управління можна здійснювати як для індивідуальних потоків, так і для згрупованих за певними ознаками: джерелі, призначення, додатку або будь-якої їх комбінації. Це надає широкі можливості для QoS або гарантованого обслуговування для певних програм [3].

Рис. 1.10 Виділення потоку трафіку, на основі загальних ознак

Контролер статично або динамічно програмує і встановлює правила відповідності для потоків. Правила потоку визначають базові інструкції, які регулюють пересилання, зміна або видалення кожного пакета, який потрапляє в SDN-комутатор [1-2][7]. Установка правил передачі пакетів контролером для нових потоків може здійснюватися двома способами [2-3][8]:

Реактивний - пристрої передачі запитують контролер, а контролер формує одне правило і встановлює його на пристрій передачі, що відправило запит.

Проактивний - контролер передає політики по ієрархії пристроїв передачі таким чином, що необхідність в контролері для обробки нових потоків майже не виникає.

Залежно від режиму управління, топологія з централізованим управлінням може мати різні вразливості безпеки, які будуть розглянуті у другому розділі.

На сьогоднішній день на ринку представлено безліч SDN контролерів таких як: OpenDayLight ONOS Floodlight Runos і ін. Але всі вони повинні відповідати основним вимогам (табл.1.2), сформованим в ході розробки і впровадження котролерів в експлуатацію.

З усього вищесказаного випливає, що контролер виступає в ролі елемента реагування: отримує повідомлення від комутаторів по каналах управління і виробляє відгуки, які змінюють вміст таблиць комутації. За рахунок логічно централізованого інтелекту в програмному SDN контролері, мережа для додатків і політик стає як єдиний логічний комутатор.

Основні вимоги до контролера представлені в таблиці 2.

Таблиця 1.2

Основні вимоги до контролера

Гнучкість	Масштабованість	Тимчасова розширюваність	Продуктивність
- сумісність API з використанням спільних структур і моделей програмування; - системна інтеграція і об'єднання додатків на організовані робочі процеси більш високого рівня	- архітектура повинна дозволяти створювати плагіни незалежно один від одного і інфраструктур и контролера, - мінімальний час системної інтеграції підтримка кластеризації	- інфраструктура контролера адаптивна до схем даних (моделям), з динамічно завантажуваних плагінів або пристроїв	- стійкість до різних видів навантаження - для підтримки розробки SDN додатків забезпечувати зв'язок із середовищем розробки додатків

Висновки

1. Проведено огляд технології програмно-конфігурованих мереж, розглянута архітектура SDN.

2. Описаний відкритий стандарт протокол OpenFlow та мережевий пристрій, який підтримую роботу по протоколу OpenFlow – комутатор OpenFlow.

3. Описані компоненти OpenFlow комутатора, формат записів в таблиці мережевих протоколів, процес обробки на основі таблиці, конвеєрна обробка пакета.

4. Розглянутий віртуальний комутатор Open vSwitch, описана структура комутатору.

5. Приведені основні функції та вимоги до контролера OpenFlow.

РОЗДІЛ 2

ІСНУЮЧІ РІШЕННЯ ДЛЯ РОЗРОБКИ ДОДАТКІВ ЯКІ ВИКОРИСТОВУЮТЬСЯ В SDN

2.1 ОСНОВНІ ФУНКЦІЇ КОНТРОЛЕРА SDN

Як ми вже згадували раніше, SDN розділяє площину даних та площину управління. По суті, інтелект мережі переміщується до контролера; всі обчислення виконуються там, і багато додатків та функцій можна додати за потреби. Основні модулі обговорюються в [9], де пропонується легкий контролер несучої якості. До основних модулів належать: модуль виявлення посилань, модуль топології, модуль зберігання даних, модуль створення стратегій, модуль таблиці потоків та модуль керуючих даних. По суті, два модулі відповідають за надання послуги маршрутизації: менеджер топології та модулі пошуку посилань. Модуль Link Discovery відповідає за виявлення та підтримку стану фізичних посилань у мережі. Існує два типи виявлення посилань: виявлення посилань між вузлами OpenFlow (перемикачі OpenFlow) за допомогою традиційного протоколу виявлення шарів посилань, LLDP (802.1AB) та виявлення посилань між вузлом Edge OpenFlow і хостом. Ця процедура запускається контролером, коли будь-який невідомий трафік надходить у домен OpenFlow. Таким чином, інформація, зібрана модулем Link Discovery, використовується для побудови бази даних сусідів у контролері, що фіксує всіх сусідів OpenFlow. Тому менеджер топології створює та підтримує інформацію про топологію в контролері та обчислює маршрути в мережі. Цей модуль використовує базу даних сусідів для обчислення топологій мережі на основі отриманої інформації з модуля Link Discovery. Менеджер топологій створює глобальну базу даних топології на контролері, яка містить інформацію про найкоротший (і альтернативний) шлях до будь-якого вузла або хосту OpenFlow.

2.2 ОСОБЛИВОСТІ КОНТРОЛЕРА SDN

1) Мова програмування

Запуск крос-платформенного, багатопотокового, легкого в освоєнні, швидкого доступу до пам'яті та належного управління пам’яттю є основними характеристиками мов програмування. Вибираючи певний контролер, ми повинні враховувати ці фактори, оскільки вони впливають на його ефективність та швидкість розробки. Python, C ++ та Java - найбільш використовувані мови для програмування контролерів SDN. Взагалі, кодовані контролери Java мають характеристику для роботи на різних платформах і мають хорошу модульність, контролери, кодовані C, забезпечують високу продуктивність, але не мають високої модульності, хорошого управління пам’яттю та хорошого графічного інтерфейсу, а кодовані Python не мають реальної багатопотокової обробки.

2) Підтримка OpenFlow

Протокол OpenFlow є ключовим фактором, що сприяє програмно визначеним мережам. Це був перший стандартизований інтерфейс на південь. Це дозволяє безпосередньо маніпулювати площиною пересилання перемикачів OpenFlow [10]. Вибираючи контролер OpenFlow, ми повинні розуміти функціональність OpenFlow, яку підтримує контролер, а також план розвитку для впровадження нових версій OpenFlow, таких як v1.3 або v1.4. Однією з причин необхідності враховувати це є те, що така важлива функціональність, як підтримка IPv6, наприклад, не є частиною OpenFlow v1.0, а є частиною стандарту OpenFlow v1.3.

3) Мережева програмованість

Мережева програмованість - це найважливіша перевага введення SDN для вирішення безпрецедентних складностей управління в сучасній мережі із вибухом кількості підключених пристроїв та розгортанням нових служб. Використання парадигми «від пристрою до пристрою» для управління масштабними майбутніми мережами буде неможливим. Старий статичний спосіб управління мережевими пристроями займає багато часу, спричиняє

помилки та призводить до невідповідностей. Програмно визначена парадигма приховує ці труднощі управління, вводячи автоматизацію та динамічність у процесі управління. Автоматизовані сценарії можна запускати через CLI, а додатки можна розгортати поверх платформи контролера для виконання заздалегідь визначених завдань та функцій управління. Підтримка контролера програмованості мережі в основному залежить від ступеня інтеграції великої кількості північних інтерфейсів, хорошого графічного інтерфейсу користувача та інтерфейсу командного рядка (CLI) [11].

4) Ефективність (продуктивність, надійність, масштабованість та безпека)

Ефективність контролера - це загальний термін, який використовується для позначення різних параметрів - продуктивності, масштабованості, надійності та безпеки. Різні показники, такі як кількість інтерфейсів, які може обробляти контролер, затримка, пропускна здатність тощо визначають те, що ми називаємо продуктивністю. Так само існують різні показники, що визначають масштабованість, надійність та безпеку. Більшість робіт, проведених для порівняння контролерів, враховують лише критерії ефективності. Крім того, централізація управління в схемі SDN представлятиме серйозну проблему з точки зору надійності та продуктивності. Таким чином, розподілена схема, підтримувана деякими контролерами, має на меті впоратися з цією проблемою [12].

5) Southbound Interfaces

API Southbound дозволяє ефективно контролювати мережу. Ці API використовуються контролером для динамічного внесення змін до правил переадресації, встановлених на пристроях площини даних, що складаються з: комутаторів, маршрутизаторів і т. Д. Хоча OpenFlow є найбільш відомим з протоколів SDN для API на південь, це не єдиний один доступний або в розробці. NETCONF (стандартизований IETF), OF-Config (підтримується Open Network Foundation (ONF)), Opflex (підтримується Cisco) та інші є прикладами інтерфейсів на південь, що використовуються для управління

мережевими пристроями. Крім того, деякі протоколи маршрутизації, такі як IS-IS, OSPF, BGP, також розробляються як інтерфейси на південь у деяких контролерах з метою підтримки гібридних мереж (SDN та не SDN) або застосування традиційних мереж у програмному забезпеченні способом [13].

6) Northbound Interfaces

Northbound Interfaces використовуються прикладним рівнем для зв'язку з контролером. Вони є найбільш важливою частиною в архітектурі контролера SDN, оскільки цінність SDN пов’язана з інноваційними програмами, які вона потенційно може підтримувати та активувати. Оскільки вони настільки важливі, API, що приєднуються до півночі, повинні підтримувати широкий спектр програм. Ці API дозволяють також з'єднуватися з автоматизованими стеками, такими як OpenStack або CloudStack, що використовуються для управління хмарою. Нещодавно Open ONF зосередився на API SDN на північ після роботи зі стандартизації інтерфейсу на південь (OpenFlow). Вони створили робочу групу, яка буде писати код, розробляти прототипи та шукати створення стандартів [14]. В даний час протокол REST (Представницький державний трансфер), здається, є найбільш використовуваним інтерфейсом на північ, і більшість контролерів реалізують його.

7) Партнерство

Перебуваючи під наглядом належного партнерства, контролер SDN матиме шанси на тривалий час підтримувати та вдосконалювати [11]. Досвід роботи в мережі та комп’ютерних доменах, а також економічний потенціал організації партнера є основними критеріями упередження довіри та використання продуктів. Cisco, Linux Foundation, Intel, IBM, Juniper тощо - це приклади надійних організацій, які виходять на ринок SDN та беруть участь у розробці контролерів.

За попередні два роки було проведено кілька опитувань [15-21], які надали нам списки найбільш відомих контролерів. По суті, більшість перелічених функцій враховуються при порівнянні контролерів.

2.3 Контролер OpenDayLight

Одним з контролерів з підтримкою OpenFlow є OpenDayLight контролер. OpenDayLight - Open-source проект з гнучкою модульної платформою MD-SAL для вбудовування різних плагінів з підтримкою декількох протоколів і забезпечення узгоджених послуг для модулів і додатків. Модельний рівень абстракції сервісу (MD-SAL) - це середовище OpenDayLight, для створення нових функцій у вигляді служб і драйверів протоколів. SAL надає такі базові сервіси, як Device Discovery (виявлення мережевих пристроїв в мережі і підключення до них), який в свою чергу використовуються такими модулями як Topology Manager (зберігання, оновлення інформації про топології мережі) і т.д.

Проект розробляється під ліцензією EPL v1.0 (Eclipse public license) за підтримки Linux Foundation. Розробляється на мові Java і є кросплатформним [22].

Контролер OpenDayLight реалізований виключно в програмному забезпеченні і зберігається в його власній віртуальній машині Java (JVM).

OpenDayLight побудований з пакетів OSGi і платформи збірки - Java Karaf на основі інфраструктури Apache Felix Framework або Eclipse Equinox OSGi використовує Maven як інструмент збірки.

Apache Karaf забезпечує установку функцій Karaf, і входить в програмне забезпечення платформи OpenDayLight. За замовчуванням OpenDayLight не має попередньо встановлених функцій.

OSGi - Java-специфічне середовище, яка покращує спосіб взаємодії Java-класів в рамках однієї JVM. Використовується для запуску додатків і дозволяє динамічно підключати плагіни для використання і зв'язку різних Southbound протоколів.

Як Karaf, так і OSGi забезпечують певний рівень ізоляції з явними межами коду, імпортом пакетів, експортом пакетів і іншими функціями, пов'язаними з безпекою.

Архітектурні принципи OpenDayLight:

- модульність і розширюваність - модульний, розширюваний контролер підтримує установку, видалення і оновлення використовуваних служб, під час роботи (без виключення);

- множинна підтримка Southbound протоколів - використання різних мережевих протоколів (OpenFlow, BGP і т.д.) через плагіни SB протоколу, забезпечення єдиного набору послуг і API для додатків через загальний набір NB API-інтерфейсів (Rest API, OSGi framework);

- службовий абстрактний рівень (Service Abstraction Layer) - використання різних Northbound протоколів;

- consistent clustering (розподілена версія) - надає відмовостійкість і підтримку розподіленої версії (High Avalability Model), дозволяє розгортати контролер на декількох фізичних серверах, гарантуючи тим самим відмовостійкість і балансування навантаження;

- роздільність - управління різними частинами мережі, за допомогою різних компонентів контролера.

Архітектура OpenDayLight представлена на рисунку 2.1.

Рис. 2.1 Архітектура OpenDayLight

Останній реліз Oxygen повністю підтримує протокол OpenFlow 1.3, BGP-LS, протокол OVSDB, PCEP, SNMP. Є додаткова підтримка OVSDB

protocol, отже, може використовувати всі особливості і розширення Open vSwitch [22].

Функціональний огляд

The OSGi framework дозволяє динамічно підключати плагіни для використання нових Southbound протоколів. SAL надає такі базові сервіси, як Device Discovery (виявлення мережевих пристроїв в мережі і підключення до них), кото-рий в свою чергу використовуються такими модулями як Topology Manager (зберігання, оновлення інформації про топології мережі) і.т.д. Грунтуючись на запиті сервісу - SAL маппірует потрібний плагін і використовує найбільш підходящий southbound протоколі.

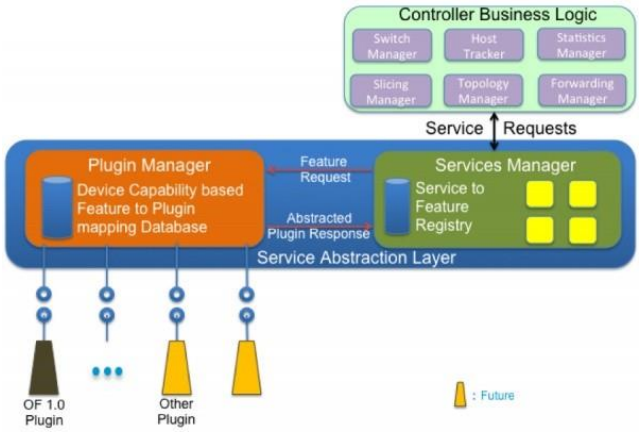

Рис. 2.2 структура OpenDaylight(www.opendaylight.org)

Одним з важливих переваг OpenDaylight є підтримка розподіленої версії (High Avalability Model), дозволяє розгортати контролер на декількох фізичних серверах, гарантуючи тим самим відмовостійкість і балансування навантаження.

2.4 POX-контролер.

POX контролер - open-source проект, що розробляється на Python під ліцензією GNU License (офіційно підтримує Windows, Mac OS, Linux)

POX зараз підтримує OpenFlow 1.0 (частково підтримує під Openflow 1.1), також є підтримка Open vSwitch / Nicira extensions (спеціальне розширення протоколу OpenFlow, що дозволяє створювати більш гнучкі налаштування маршрутизаторів, приклад: action = normal - flow який змушує

обробляти співпав пакет, як звичайний L2 Linux bridge, ставити числові мітки на пакети, які йдуть по forwarding pipeline і таким чином контролювати обробку).

POX сам по собі ніяк не керує трафіком - функціональність контролера з- здается запуском окремих компонент разом з контролером (components), следова- тельно цільова аудиторія проекту POX - розробники, які хочуть реалізувати свої проекти. Стандартний дистрибутив включає в себе набір базових компонент.

Розробка компонент

Розробка компонент POX компоненти по суті це модулі, написані на Python, отже в кон- Троллер можна додати будь-який код, який розробник захоче. За угодою, кожна компонента містить спеціальну функцію, яка виконується при за- пуску контролера:

```
def launch (args ..):
// код компоненти
```

Стандартно, кожна компонента може бути запущена тільки один раз, але спеці альних командами можна запустити одну і ту ж компоненту в декількох екзем- Пляра.

POX API

POX містить об'єкт - "ядро (core)", який реалізує основу POX's API. Основ- ная мета даного об'єкта - комунікація між запущеними компонентами. При запуску, зазначені компоненти реєструються на цьому об'єкті і всілякі пересилання між компонентами реалізовані через його використання. Якщо необхідно "зареєструвати" компоненту в "ядрі" необхідно виконати наступні дві команди:

core.register(component_class) core.registerNew(component_class)

У підсумку "ядро" (core) і стандартні компоненти бібліотеки pox надають на- дующее API's:

• Dependency and Event Management (елементи можуть бути події, а інші компоненти на них реагувати)

• Working with packets and addresses (парсинг пакетів, створення Openflow пакетів з "нуля")

• Using threads, timers, tasks (POX's recoco library спеціальна бібліотека для многопоточности; наприклад потрібно збирати статистику з контролера каж- Диє N секунд)

• Working with asynchronous sockets (комунікація між контролером і уда- ленним хостом)

• Openflow protocol management (розробка Openflow додатків)

Загальний огляд

POX в основному використовується для досліджень ПКС, тому що надає простий і зручний Northbound API для невеликих завдань. Основна мета цього проекту - розробка і дослідження парадигми ПКС, для полнценних комерційних додатків цей проект не підходить через обмежений API і повільної роботи (т.к. Написаний на Python)

2.5 Floodlight-контролер.

Контролер розробляється на Java 1.7+ під ліцензією Apache 2.0. Поддер- живає тільки топології без циклів між різними компонент зв'язності. Має модульну структуру, тобто перед запуском в спеціальний конфігураційний файл додаються назви сервісів, які будуть використані. Підтримує наступні сервіси:

• Forwarding - стандартний модуль, що забезпечує реактивне конфігурація OpenFlow маршрутизаторів.

• Static Flow Entry Pusher - спеціальний додаток для для додавання flow-записів в обраний маршрутизатор.

• Firewall (брандмауер) - застосування ACL правил, для обмеження трафіку (налаштовується через Rest API)

• Learning Switch - звичайний L2 маршрутизатор (можлива настройка через Rest API)

• Load Balancer (балансувальник навантаження) - балансування tcp, ping, udp трафіку.

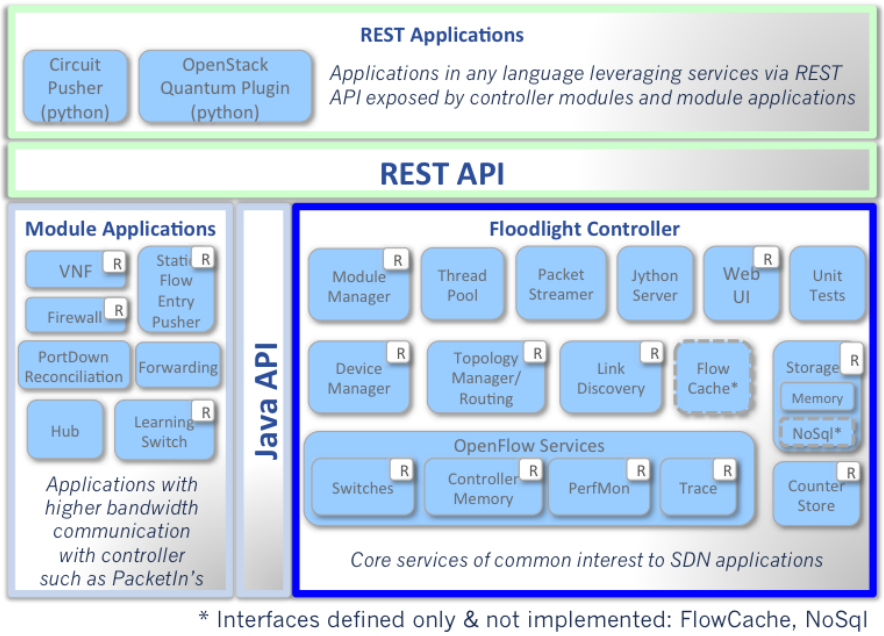

Рис. 2.3 Структура Floodlight (www.docs.projectfloodlight.org)

На зображенні показана архітектура контролера. По суті контролер виконує стандартні функції сканування мережі, підключення нових пристроїв, робота з OpenFlow (1.1), наявність графічного інтерфейсу.

Для розробки додатків на базі даного контролера, надається Restful Java API.

Підсумки огляду

OpenDaylight controller величезно швидко розвивається проект спонсорується такими великими організаціями як: Cisco, IBM, Red Hat. З переваг можна виділити: працює на JVM, тобто працює всюди, де є Java; модульність - працює з багатьма Northbound APIs (Rest API, OSGi frameowork) і Southbound APIs які між собою незалежні (важлива особливість, що модулі можна завантажувати прямо під час роботи

контролера. Є додаткова підтримка OVSDB protocol, отже може використовувати всі особливості і розширення Open vSwitch.

Таблиця 2.1

Огляд контролерів

Проект Характеристика	POX	Floodlight	OpenDaylight
Мова	Python 2.6	Java 1.7+	Java 1.7+
рядків коду	56961	90578	332317
підтримка OVS	1.11	2.0+	2.0+
OpenFlow Hardware	Да	Да	Да
Тип API	Rest API	Rest API	Rest API, OSGi framework
Підтримка GRE	Нет	Да	Да
Розподілена модель	Да	Да	Да

В цілому можна сказати, що POX контролер використовується для перевірки ефективості ПКС на різних топологіях мереж і тестування протоколу OpenFlow. Floodlight і OpenDaylight можна об'єднати в одну категорію, як два проекти, що розробляються на мові Java. Хоча Floodlight можна назвати перехідним між POX і OpenDaylight, тому що останній підтримує будь-які топології мереж, велика кількість протоколів маршрутизації, є більш гнучким і краще підходить для вирішення реальних бізнес-завдань.

Серед розглянутих проектів сильно виділяється OpenDaylight. Є підтримка розподіленої моделі, OSGi фреймворк дозволяє завантажувати модулі без перезавантаження контролеру, активна підтримка Linux спільноти (спонсори: IBM, Cisco, Nicira, OpenFlow Network Foundation і т.д) Додано багато інтерфейсів на Southbound (HTTP, COAP, PCEP, LACP, OpFlex, SNMP та ін.) Та впроваджені нові модулі (посередник даних IoT (IoTDM), уніфікований захищений канал USC тощо). Таким чином, це перший контролер, що входить у домен IoT. Підтримуючи широкий спектр

Southbound інтерфейсів та розподіленої парадигми управління, здається, він є контролером Інтернету майбутнього. Підтримка плагіна OpenStack Neutron в його архітектурі також має надзвичайно важливе значення при розгортанні програмно визначених країв у відповідь на поширення обчислень Edge.

Успадковуючи потужність Java / Javascript у програмуванні графічних інтерфейсів, він представляє хорошу функцію графічного інтерфейсу. Перебуваючи у партнерстві з відомими постачальниками мереж та дослідницькими спільнотами, вони мають чіткий план розвитку та належну документацію. Крім того, підтримка розподіленої схеми робить його здатним здійснити справжнє розгортання SDN.

Висновки

1. Описані основні функції контролера SDN та особливості контролера SDN такі, як мова програмування, підтримка OpenFlow, мережева програмованість, ефективність, southbound interfaces, northbound interfaces, партнерство.
2. Розглянуті найбільш популярні контролери такі, як OpenDayLight описана архітектура, POX контролер описана POX API, Floodlight контролер описана структура.
3. Приведені підсумки та призначення цих контролерів та прийнято рішення використовувати OpenDayLight.

РОЗДІЛ 3

РОЗРОБКА СЕГМЕНТУ МЕРЕЖІ SDN

3.1 Розгляд Mininet

Для тестування додатків контролера можна використовувати: фізичне обладнання, віртуальне обладнання (емуляція), моделювання. Перший підхід має високим ступенем довіри, але при цьому є вкрай не зручним при експериментуванні (зміні топології, коду, налаштувань). Емуляція покриває цей недолік і при цьому дозволяє заощадити на придбанні обладнання. Але через те, що кожен пристрій вдає із себе віртуальну машину, при великих мережевих топологіях потрібні значні обчислювальні потужності. Найоптимальнішим інструментом для тестування є моделювання, яке використовує низьку вимога до ресурсів.

Віртуальні мережі повинні бути:

- гнучкими, здатними підтримувати різні топології, архітектури маршрутизації і переадресації, а також незалежні конфігурації;
- керованими, щоб відокремлювати політику, яку намагається вказати мережевий оператор, від тих механізмів, як ці політика реалізується;
- масштабованими, для збільшення кількості співіснують віртуальних мереж;
- захищеними, шляхом виділення різних логічних мереж друг від друга;
- програмованими;
- гетерогенними, для підтримування різних технологій.

Віртуальні мережі складаються з двох компонентів: вузлів і ребер. Самі фізичні вузли повинні бути віртуалізувати. Один з можливих способів віртуалізації вузла являє собою віртуальну машину. Цей спосіб віртуалізації вузла використовує віртуальне середовище, таку як VServer або Jail. Гіпервізор або інша технологія дозволяє віртуальному середовищі ефективно

надавати фізичне обладнання, щоб забезпечити ілюзію декількох гостьових вузлів. Приклади віртуалізації вузла: середа віртуальної машини, таких як Xen або VMware; віртуалізація на рівні ОС або віртуальні середовища, таких як Linux VServer. У віртуальній мережі необхідно з'єднати ці віртуальні машини. Кожна віртуальна машина або віртуальне середовище має свій власний вид мережевого стека. Потрібно забезпечити видимість того, що ці вузли з'єднані один з одним через 2-й рівень OSI, навіть якщо вони насправді розділені декількома IP-мережами. Один з можливих способів зробити це полягає в інкапсуляції Ethernet-кадру коли він залишає межі VM в IP-пакет. У пункті призначення IPпакет де-инкапсулируется і передає вихідний Ethernet-кадр до ВМ або віртуальному середовищі. Кожен з однієї цих фізичних хостів, може насправді, розмістити кілька віртуальних машин або віртуальних середовищ, що створює необхідність у віртуальному комутаторі, який знаходиться на фізичному вузлі. Цей віртуальний комутатор з'єднує в віртуальну мережу 2-го рівня OSI все ВМ. Linux-міст, OpenVSwitch є прикладами програмних комутаторів, які можуть виконувати цей тип функції.

Починаючи з версії 2.6.24 ядром Linux підтримуються механізми віртуалізації і ізоляції - Cgroups, які дозволяють забезпечити мережевими інтерфейсами, таблицями маршрутизації і ARP-таблицями процеси в рамках однієї операційної системи. Це один з видів віртуалізації на рівні ОС, що дозволяє запустити безліч однотипних процесів в ізольованому і обмеженому по ресурсів оточенні. Дану технологію використовує середовище для моделювання MiniNet, яка дозволяє створити віртуальну мережу на малопотужному обладнанні. При запуску Mininet за допомогою сценарію «mn», кожен хост в віртуальної мережі є bash-процес с власною мережею простору імен, фактично це віртуалізація на рівні ОС (рис.3.1).

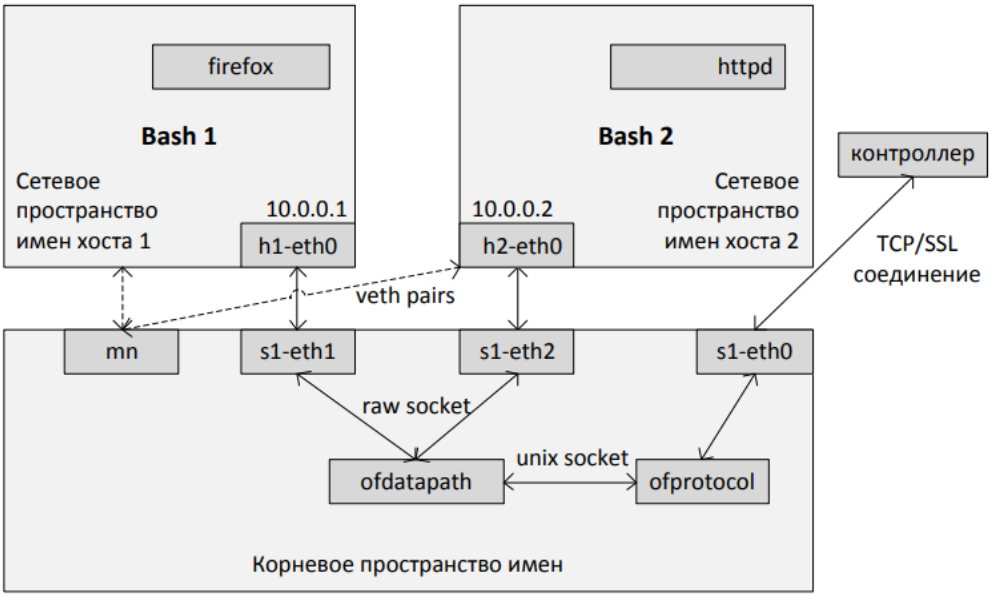

Рис.3.1 Архітектура MiniNet

Так, кожен з цих віртуальних вузлів, мають свій власний мережевий стек. Кореневе простір імен управляє зв'язком між цими різними віртуальними вузлами, а також виконує роль комутатора, який з'єднує ці вузли в створену вами топологію. Віртуальним парам Ethernet призначені два простору імен. Наприклад, S1 eth1 призначається інтерфейс в мережевому просторі H2, S1 eth2 присвоюється мережевого простору імен H3. Комутатор OpenFlow ефективно виконує переадресацію між інтерфейсами в кореневому просторі імен. Але оскільки інтерфейси є парними, ми отримуємо ілюзію відправки трафіку між h2 і h3. Коли ми вносимо зміни в комутаторі OpenFlow через контролер, ми фактично це робимо в кореневому просторі імен. Подібні техніки дозволяють Mininet створювати в просторі ядра або користувача комутатори, OpenFlow-контролери і хости і взаємодіяти в рамках модельованої мережі. Як віртуальних комутаторів використовується адаптована реалізація Open vSwitch.

Mininet - це мережевий емулятор, який має змогу створювати мережу віртуальних хостів, комутаторів, контролерів і посилань. Хости Mininet працюють під управлінням стандартного мережевого програмного

забезпечення Linux, а його комутатори підтримують OpenFlow для дуже гнучкої маршрутизації, що налаштовується, і програмно визначена мережа.

Mininet підтримує дослідження, розробку, навчання, створення прототипів, тестування, налагодження та інші завдання, які можуть бути корисними, якщо у вас є повна тестова мережа на ноутбуці або іншому ПК.

Мінінет:

- Надає простий і недорогий мережевий випробувальний стенд для розробки додатків OpenFlow;

- Дозволяє одночасно, окремо, працювати декільком розробникам над однією і тією ж топологією;

- Підтримує регресивні тести системного рівня, які можна повторювати і легко упаковувати;

- Забезпечує комплексне тестування топології без необхідності підключення до фізичних мереж;

- Включає інтерфейс командного рядка з підтримкою топології і OpenFlow для налагодження або запуску загальномережевих тестів;

- Підтримує довільні кастомні топології і включає в себе основний набір параметрезованих топологій;

- Надає простий і розширюваний API Python для створення мереж і експериментів.

Mininet надає можливість отримувати правильну поведінку системи(на скільки дозволяє ваше обладнання, продуктивність), та дає можливість поекспериментувати з топологіями.

У мережах Mininet працює реальний код, який включає в себе стандартні мережеві додатки Unix / Linux, а також реальне ядро Linux і мережевий стек (та містить будь-які розширення ядра, до яких ви маєте доступ, звичайно якщо вони суміщуються з мережевими просторами імен).

З цієї причини код, який ви розробляєте і запускаєте на Mininet, для контролера OpenFlow, модифікованого комутатора чи хоста, можна перенести в реальну систему з мінімальними змінами для тестування, оцінки

продуктивності та розгортання в реальних умовах. Важливо те, що проект, який працює в Mininet, зазвичай може переходити безпосередньо до апаратних комутаторів з великою ймові для пересилання пакетів на лінійній швидкості.

Mininet суміщує в собі найкращий функціонал з усіх емуляторів, та апаратних випробувальних стендів і симуляторів.

У порівнянні з підходами, які засновані на повній віртуалізації системи, Mininet:

- Завантажується швидше: секунди замість хвилин;

- Масштабування більше: сотні хостів та комутаторів проти однозначних чисел;

- Має велику пропускну здатність: зазвичай загальна пропускна здатність 2 Гбіт / с на скромному обладнанні;

- Легко встановлюється: доступна попередньо упакована віртуальна машина, яка працює на VMware або VirtualBox для Mac / Win / Linux з уже встановленими інструментами OpenFlow v1.0.

Mininet має переваги перед апаратними тестовими системами, а саме:

- недорого і завжди доступні (навіть до конференції терміни);

- швидко змінює налаштування і перезапускається;

Перед симуляторами, Mininet має такі переваги :

- запускає реальний немодифікований код, який включає код додатка, код ядра ОС і код площини управління (код контролера OpenFlow та код Open vSwitch);

- легко интегрується та підключається до справжніх мереж;

3.2 Огляд технології VirtualBox

Розробка сегмента SDN проводилася с допомогою технології віртуалізації - Oracle VM VirtualBox (VB). VirtualBox - це універсальний повнофункціональний віртуалізатор для обладнання x86, призначений для використання на сервері, настільному комп'ютері та вбудованих системах [25].

Таблиця 3.1

Огляд можливостей Oracle VM VirtualBox

Переносимість	Oracle VM VirtualBox працює на великій кількості 32-бітних і 64 бітних хост-ОС. Це так званий розміщений гипервизор, іноді званий гіпервізором типу 2. У той час як гипервизор типу 1 буде працювати безпосередньо на обладнанні, Oracle VM VirtualBox вимагає установки існуючої ОС. Таким чином, він може працювати разом з існуючими додатками на цьому хості.
Не потрібна апаратна віртуалізація	Для багатьох сценаріїв Oracle VM VirtualBox не вимагає функцій процесора, вбудованих в нове обладнання, таке як Intel VT-x або AMD-V. На відміну від багатьох інших рішень для віртуалізації, ви можете використовувати Oracle VM VirtualBox навіть на старому обладнанні, де ці функції відсутні.
Гостьові доповнення: загальні папки, безшовні вікна, 3D віртуалізація	Гостьові доповнення Oracle VM VirtualBox є пакетами програмного забезпечення, які можна встановлювати всередині підтримуваних гостьових систем для підвищення їх продуктивності і забезпечення додаткової інтеграції та зв'язку з хост-системою. Після установки Guest Additions віртуальна машина буде підтримувати автоматичне налаштування дозволів відео, безшовних вікон, прискореної тривимірної графіки і багато чого іншого.

Продовження таблиці 3.1

Багатопоколінні розгалужені знімки	Oracle VM VirtualBox може зберігати довільні знімки стану віртуальної машини. Ви можете повернутися назад в часі і повернути віртуальну машину до будь-якого такого знімка і запустити альтернативну конфігурацію віртуальної машини, створивши ефективну дерево знімків.
ВМ групи	Oracle VM VirtualBox надає функцію груп, яка дозволяє користувачеві організовувати і контролювати віртуальні машини як колективно, так і індивідуально. На додаток до базових груп будь-яка віртуальна машина також може знаходитися в більш ніж одній групі, а групи можуть бути вкладені в ієрархію. Це означає, що ви можете мати групи груп. Як правило, операції, які можна виконувати з групами, аналогічні операціям, які можна застосовувати до окремих віртуальних машин: запуск, пауза, скидання, закриття (стан збереження, відправка виключення, відключення живлення), скидання збереженого стану, відображення в файлової системі, Сортувати.

3.3. Практичне використання Mininet для моделювання мережі SDN

3.3.1 Параметри запуску Mininet

За допомогою наступної команди будуть відображені параметри запуску Mininet:

```
$ sudo mn -h
```

3.3.2. Робота з хостами та комутаторами

Для створення мінімальної топології прописати наступну команду:

```
$ sudo mn
```

Топологія за замовчуванням - це мінімальна топологія, яка складається з одного комутатора ядра OpenFlow, який підключений до двох хостів, та контрольний контролер OpenFlow. Ця ж топологія також може бути вказана в командному рядку як --topo=minimal. Також інші топології доступні за замовчуванням. Усі чотири об'єкти (2 хоста, 1 комутатор, 1 основний контролер) тепер працюють у віртуальній машині.

Якщо не було передано ніякого спеціального тесту в якості параметра, з'явиться інтерфейс командного рядка Mininet.(CLI)

Для показу команд інтерфейсу командного рядка Mininet існує наступна команда:

```
mininet> help
```

Команда *nodes* видає список вузлів в мережі:

```
mininet> nodes
```

Для повної інформації про кожен вузол потрібно ввести команду:

```
mininet> dump
```

Ви побачите комутатор та два хоста.

Якщо ім’я хоста, комутатора чи контролера, буде набраним першим в CLI Mininet, то команди будуть виконуватися на цьому вузлі. Приклад команди в хостовому процесі:

```
mininet> h1 ifconfig -a
```

Ви можете побачити інтерфейси h1-eth0 і loopback (lo) хоста. Зверніть увагу, що первинна система Linux не може побачити інтерфейс (h1-eth0) при запуску ifconfig, оскільки він належить для мережевого простору імен хост-процесу.

А комутатор, навпаки працює за замовчуванням в кореневому мережевому просторі імен, тому виконання команди на «комутаторі» аналогічно запуску його зі звичайного термінала:

```
mininet> s1 ifconfig -a
```

Команда покаже інтерфейси комутатора, та підключення (eth0).

Mininet має можливість розмістити кожний хост, комутатор та контролер в окремий простір мережевих імен, але великої переваги в цюому немає, тільки якщо вам необхідно створити складну мережу з декількома контролерами.

Зверніть увагу, що віртуалізована лише мережа, тобто кожний хост-процес має змогу бачити однаковий набір процесів і каталогів. Прикладом може бути список процесів хостового процесу та у просторі імен кореневої мережі:

```
mininet> h1 ps -a
```

```
mininet> s1 ps -a
```

Результати будуть однакові.

3.3.3. Зв’язок між хостами

Переконаємося, що можемо виконати команду ping від хоста 0 до хоста 1:

```
mininet> h1 ping -c 1 h2
```

Ви побачите керований трафік OpenFlow. Перший вузол надсилає ARP-адрес для MAC-адреси другого, що викликає передачу *packet_in* повідомлення на контролер. Потім контролер надсилає *packet_out* повідомлення для розсилки широкомовного пакета на інші порти комутатора (саме в цьому прикладі - єдиний інший порт даних). Другий хост бачe запит ARP і відправляє відповідь. Ця відповідь відправляється контролеру, який відправляє її першому хосту та штовхає вхід потоку..

На разі, перший хост знає MAC-адресу другого та має змогу відправити свій запит(ping) через ICMP Echo Request. Цей запит разом з відповідною йому відповіддю від другого хоста, направляються до контролера і призводить до відправки запису потоку донизу (разом з фактичними відправленими пакетами).

3.3.4. Запуск простого веб-сервера та клієнта

В мережі створеною Mininet, команда ping не є єдиною командою, яка може бути запуститися на хості. Хости мають право запускати команди, програму, яка є в доступі для базової системи Linux (або VM) та її файлової системи. Крім того, можна ввести будь-яку bash команду, в тому числі керування роботою (&, jobs, killi т.д ..)

Для запуску простого серверу HTTP на h1 вводиться команда, яка зображена нижче, зробивши запит від h2, та вимкнемо сервер на h1:

```
mininet> h1 python -m SimpleHTTPServer 80 &
mininet> h2 wget -O - h1
...
mininet> h1 kill %python
```

Для виходу з CLI:

```
mininet> exit
```

3.3.5. Створення власної мережевої топології

За допомогою Mininet ви можете створити кастомну топологію зі своїми параметрами. Кастомна топологія створюється за допомогою коду Python, обираючи основні параметри для створення топологій. А також її можна буде використовувати для подальших експериментів.

Прикладом може бути, мережева топологія, яка була створена мною. Складається топологія з визначеної кількості хостів (h1 - hN), які підключені до одного комутатора(s1):

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

class SingleSwitchTopo(Topo):
   "Single switch connected to n hosts."
   def build(self, n=2):
      switch = self.addSwitch('s1')
      # Python's range(N) generates 0..N-1
      for h in range(n):
         host = self.addHost('h%s' % (h + 1))
         self.addLink(host, switch)
```

```
def simpleTest():
    "Create and test a simple network"
    topo = SingleSwitchTopo(n=4)
    net = Mininet(topo)
    net.start()
    print "Вивід інформації про стан підключення хостів"
    dumpNodeConnections(net.hosts)
    print "Тест зв’язку у мережі"
    net.pingAll()
    net.stop()

if __name__ == '__main__':
    # Вивід інформації
    setLogLevel('info')
    simpleTest()
```

Основні методи, функції, класи, змінні які зустрічаються в наведеному вище коді:

- Topo: базовий клас для топологій Mininet
- build (): метод перевизначення у вашому класі топології. Параметри конструктора (n) будуть передані йому автоматично Topo .__ init __ ().
- addSwitch (): додає комутатор до топології та повертає його ім'я
- addHost (): додає хост до топології та повертає ім'я хоста
- addLink (): додає двонаправлене посилання на топологію (і повертає ключ посилання). Посилання в Mininet двосторонні, якщо не зазначено інше
- Mininet: основний клас для створення та управління мережею
- start (): запускає вашу мережу

- pingAll (): перевіряє підключення, намагаючись усі вузли пінгувати один одного
- stop (): зупиняє вашу мережу
- net.hosts: всі хости в мережі
- dumpNodeConnections (): скидає з'єднання до / з набору вузлів
- setLogLevel ('info' | 'debug' | 'output'): встановити вихідний рівень

При запуску цього коду, ми отримаємо такий результат:

```
$ sudo python simpletest.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
Testing network connectivity
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 1 controllers
c0
*** Stopping 4 links
....
```

```
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
```

Також Mininet до базових поведінкових мереж, додатково має змогу забезпечити обмеження продуктивності та має можливість ізолювання через класи CPULimitedHost та TCLink.

Є декілька способів, як ці класи можливо використовувати. Один зі способів є доволі простий – це вказати класи як хост за замовчуванням та зв'язати класи / конструктори з Mininet(), а вже потім вказати параметри які потрібні в топології. (Також ви маєте можливість вказати спеціальні класи у самій топології, або створити свої конструктори вузлів та посилання і / або підкласи).

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

class SingleSwitchTopo( Topo ):
    def build( self, n=2 ):
        switch = self.addSwitch( 's1' )
        for h in range(n):
            # Each host gets 50%/n of system CPU
            host = self.addHost( 'h%s' % (h + 1),
```

```
                                cpu=.5/n )
            # 10 Mbps, 5ms delay, 2% loss, 1000 packet queue
            self.addLink( host, switch, bw=10, delay='5ms', loss=2,
                    max_queue_size=1000, use_htb=True )


def perfTest():
    "Створення мережі та простий тест"
    topo = SingleSwitchTopo( n=4 )
    net = Mininet( topo=topo,
                   host=CPULimitedHost, link=TCLink )
    net.start()
    print "Вивід інформації про стан підключення хостів"
    dumpNodeConnections( net.hosts )
    print " Тест зв’язку у мережі "
    net.pingAll()
    print "Тест пропускної здатності між h1 та h4"
    h1, h4 = net.get( 'h1', 'h4' )
    net.iperf( (h1, h4) )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    perfTest()
```

Створимо ще один приклад з кастомною топологією мережі. Вона матиме 4 хоста, та два комутатора, які будуть з’єднані між собою.

Програмний код , через який описується топологія має такий вигляд:

```
from mininet.topo import Topo
    class MyTopo( Topo ):
        "Simple topology example."
```

```
    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        Host1 = self.addHost( 'h1' )
        Host2 = self.addHost( 'h2' )
        Host3 = self.addHost( 'h3' )
        Host4 = self.addHost( 'h4' )
        Switch1 = self.addSwitch('s1')
        Switch2 = self.addSwitch('s2')
        # Add links
        self.addLink( Host1, Switch1 )
        self.addLink( Host2, Switch1 )
        self.addLink( Host3, Switch2 )
        self.addLink( Host4, Switch2 )
        self.addLink( Switch1, Switch2 )
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Після запуску програмного коду маємо наступний результат:

```
***Creating network
***Adding controller
***Adding hosts:
h1 h2 h3 h4
***Adding switches:
s1 s2
***Adding links:
```

```
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, s2) ***Configuring hosts
h1 h2 h3 h4
***Starting controller
c0
***Starting 2 switches
s1 s2
***Starting CLI:
mininet>
```

Інформацію про вузли та елементи можемо отримати за допомогою наступних команд:

Команда nodes, показує список всіх вузлів нашої топології:

```
mininet> nodes
available nodes are:
c0 x1 x2 x3 x4 c1 c2
```

Команда links, показує всі зв’язки між хостами та комутаторами. Також вона виводить інформацію про стан підключення:

```
mininet> links
h1-eth0 <-> s1-eth1 (OK OK)
h2-eth0 <-> s1-eth2 (OK OK)
h3-eth0 <-> s2-eth1 (OK OK)
h4-eth0 <-> s2-eth2 (OK OK)
s1-eth3 <-> s2-eth3 (OK OK)
```

Команда ifconfig, показує детальну інформацію про будь-який елемент в створеній мережі. За приклад, можемо розглянути елемент h1:

```
 mininet> h1 ifconfig
h1-eth0 Link encap: Ethernet HWaddr 8e: 4f: d8: 24: e1: e6
        inet addr: 10.0.0.1 Bcast: 10.255.255.255 Mask: 255.0.0.0
        inet6 addr: fe80 :: 8c4f: d8ff: fe24: e1e6 / 64 Scope: Link
```

```
        UP BROADCAST RUNNING MULTICAST MTU: 1500 Metric: 1
        RX packets: 17 errors: 0 dropped: 0 overruns: 0 frame: 0
        TX packets: 8 errors: 0 dropped: 0 overruns: 0 carrier: 0
        collisions: 0 txqueuelen: 1000
        RX bytes: 1326 (1.3 KB) TX bytes: 648 (648.0 B)

lo Link encap: Local Loopback
        inet addr: 127.0.0.1 Mask: 255.0.0.0
        inet6 addr: :: 1/128 Scope: Host
        UP LOOPBACK RUNNING MTU: 65536 Metric: 1
        RX packets: 0 errors: 0 dropped: 0 overruns: 0 frame: 0
        TX packets: 0 errors: 0 dropped: 0 overruns: 0 carrier: 0
        collisions: 0 txqueuelen: 0
        RX bytes: 0 (0.0 B) TX bytes: 0 (0.0 B)
```

Реалізована топологія має такий вигляд:

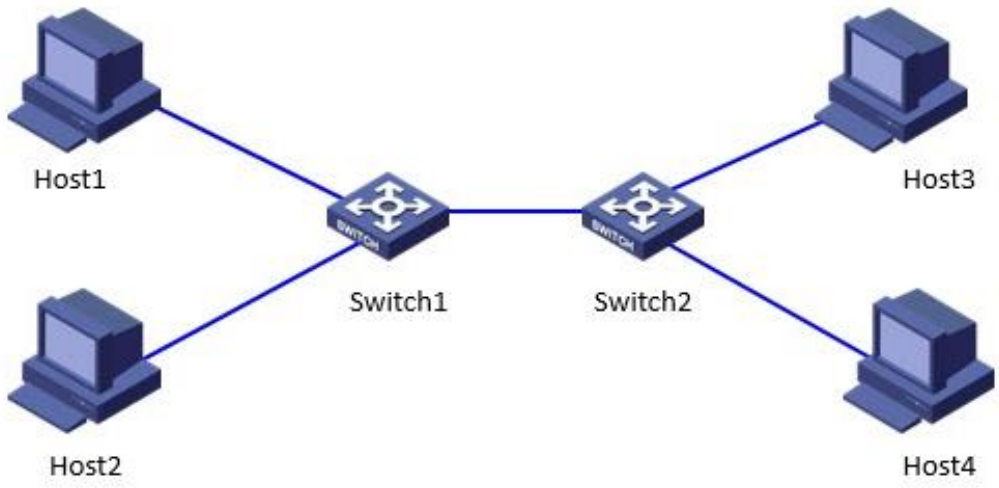

Рис 3.2. Реалізована топологія

3.4 Установка OpenDayLight на VirtualBox (Ubuntu 20.04)

3.4.1 Підготовка Ubuntu операційної системи

Після встановлення Ubuntu 20.04 потрібно оновити операційну систему, додатки та інструменти безпеки за допомогою диспетчера пакетів apt.

За допомогою функції *apt-get update*

```
$ sudo apt-get -y update
```

3.4.2 Загрузка Zip-архіва OpenDayLight

Заходимо на офіційний сайт https://www.opendaylight.org/ та переходимо у розділ DOWNLOAD HERE

Далі OpenDaylight Alluminium Zip.

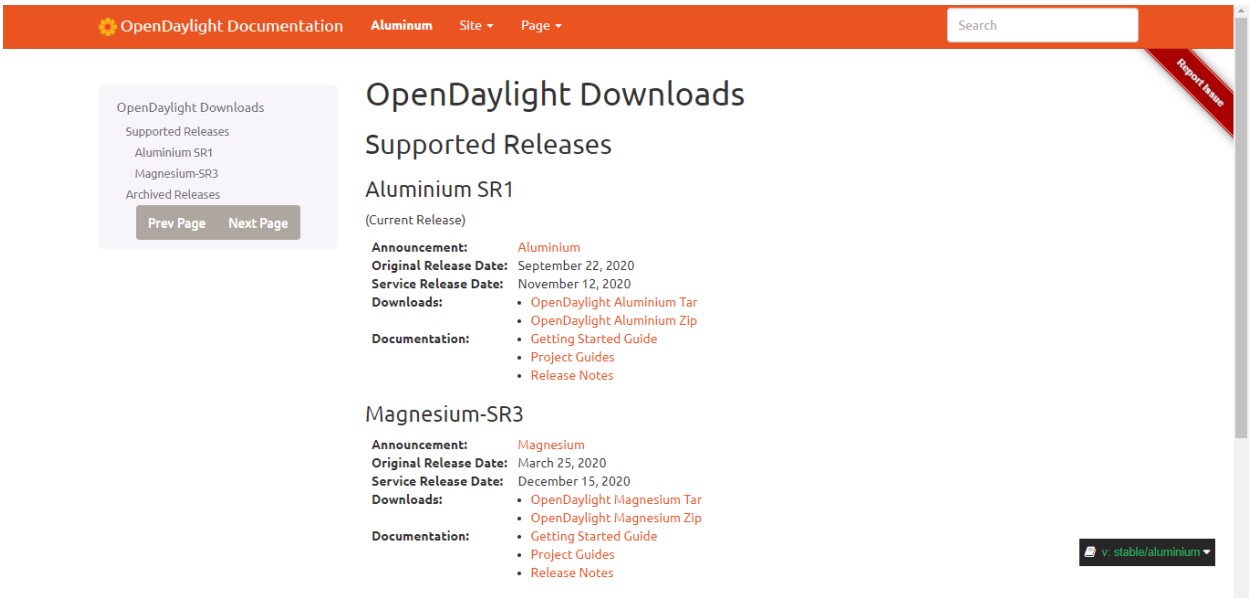

Рис 3.3. OpenDaylight Alluminium Zip

3.4.3 Добавление папки для общего доступа VM

Переходимо в налаштування віртуальної машини, у розділ загальні папки, обираємо на локальному ПК папку(в якій вже розархівований архів з контролером).

Обов’язково треба поставити позначку в check-box Авто-підключення.

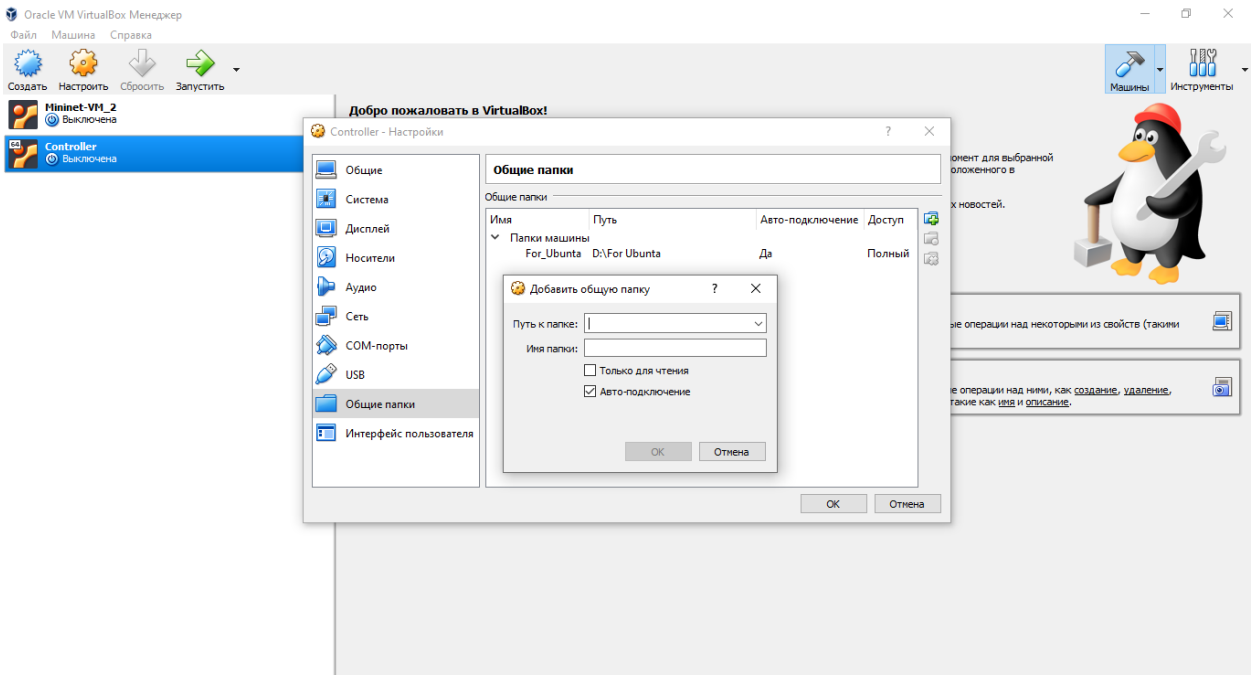

Рис 3.4.налаштування загальні папки

Результат можна побачити в провіднику системи.

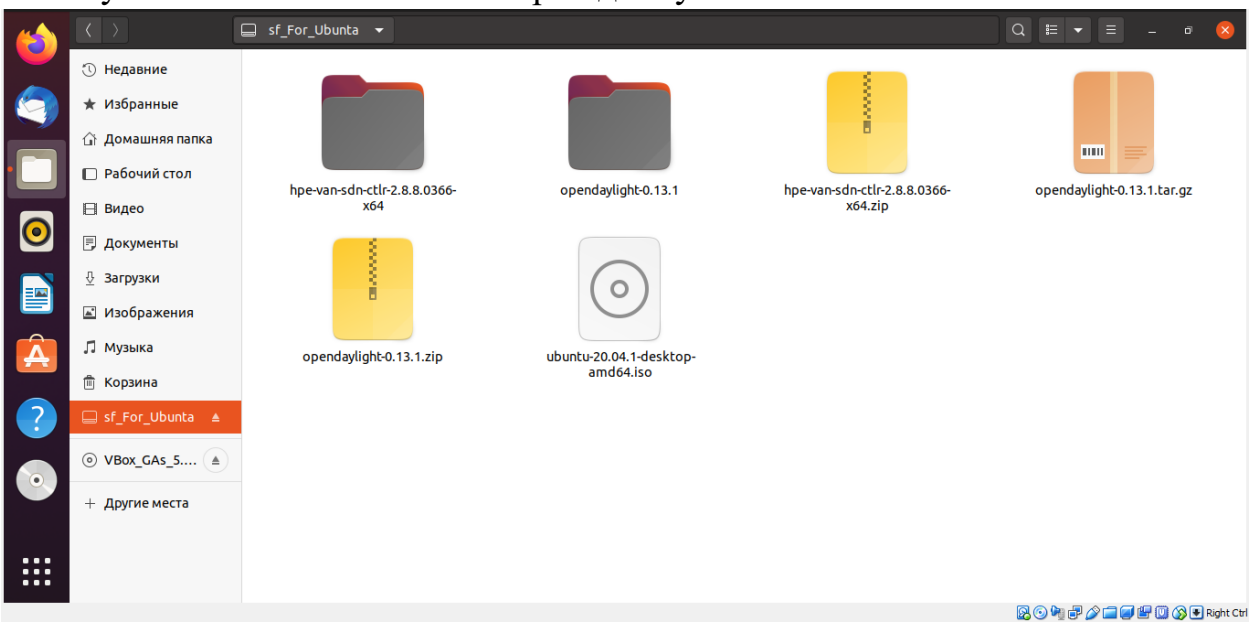

Рис 3.5. Провідник системи

3.4.4 Налаштування системи для запуску OpenDayLight

Архітектори OpenDaylight розробили OpenDaylight для екосистеми Java. OpenDaylight вимагає для роботи середовища виконання Java (JRE). OpenDaylight може використовувати або автономну JRE, або JRE, що входить в комплект Java Software Development Kit.

```
$ sudo apt-get -y install openjdk-11-jre
```

OpenDaylight хоче, щоб змінна середовища JAVA_HOME відображала розташування всього набору інструментів JAVA, а не тільки виконуваного файлу JAVA. Тому перед запуском контроллера потрібно вказати JAVA_HOME в папку з контролером.

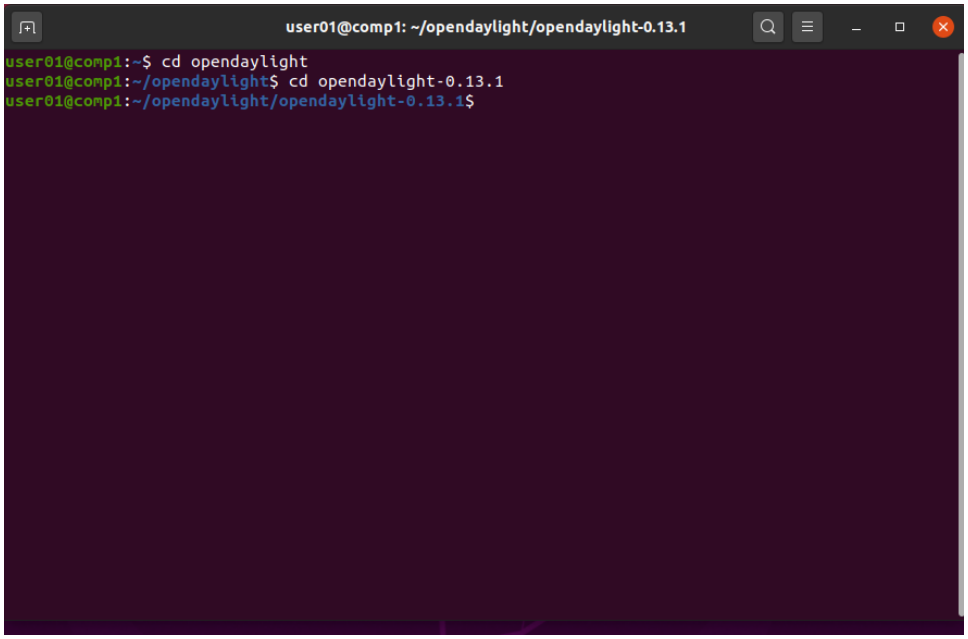

Рис 3.6.Папка с контролером

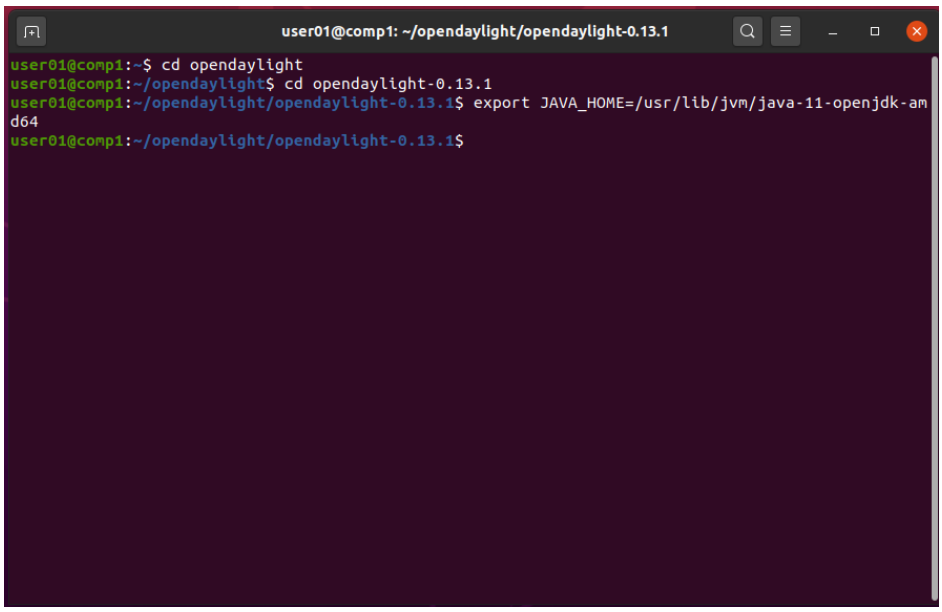

Рис 3.7. Команда вказання JAVA_HOME

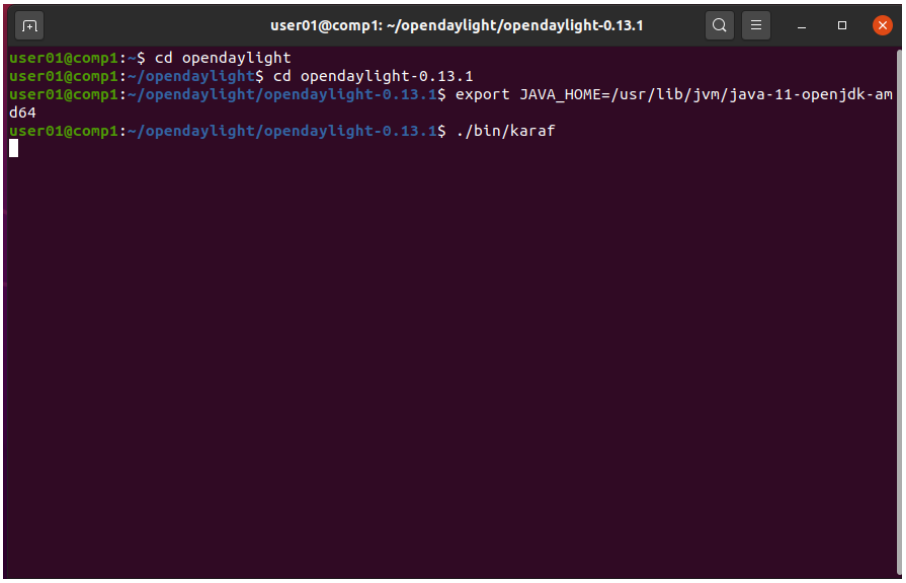

Рис 3.8. Команда запуску контролера

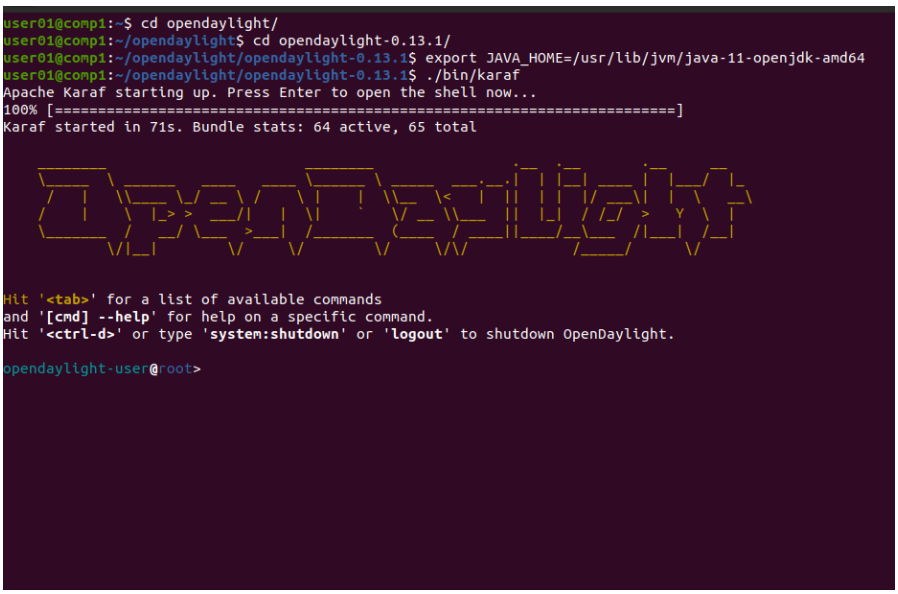

Рис 3.9. Контроллер запущен

3.4.5 Інтеграція Opendaylight в Mininet

Контролер OpenDayLight прослуховує порт 6633 OpenFlow для підключення до його вузлів. Команда netstat -a | grep 6633 побачити стан порту. Перевірка стану OpenFlow порту контролера представлена на рисунку 4.10.

Як описано вище Mininet запускає кастомні топології.

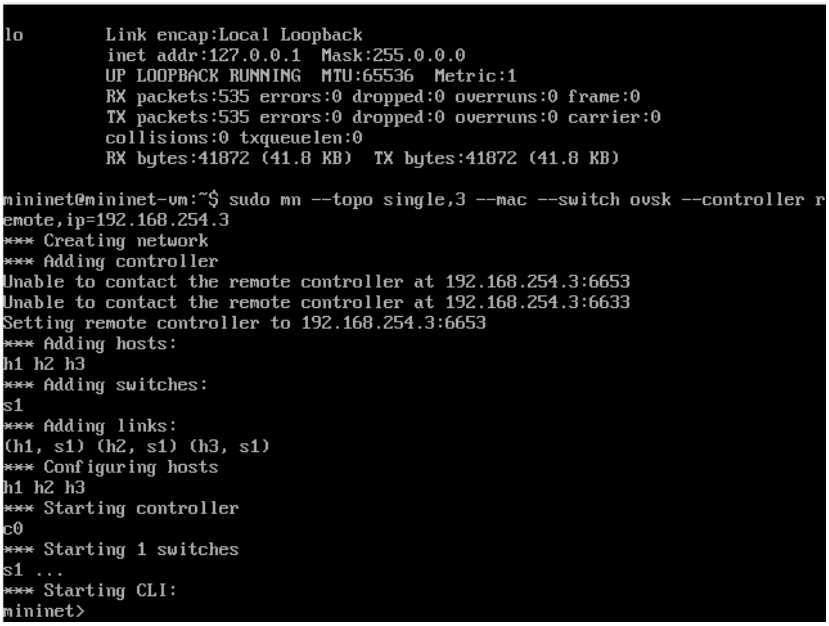

Рис 3.10. Запуск топології з зовнішнім контролером

Як бачимо емулятору не вдалося зв’язатися з пультом дистанційного керування за адресою 192.168.254.3:6653 та 192.168.254.3:6633.

Тому подивимося які порти слухає контролер:

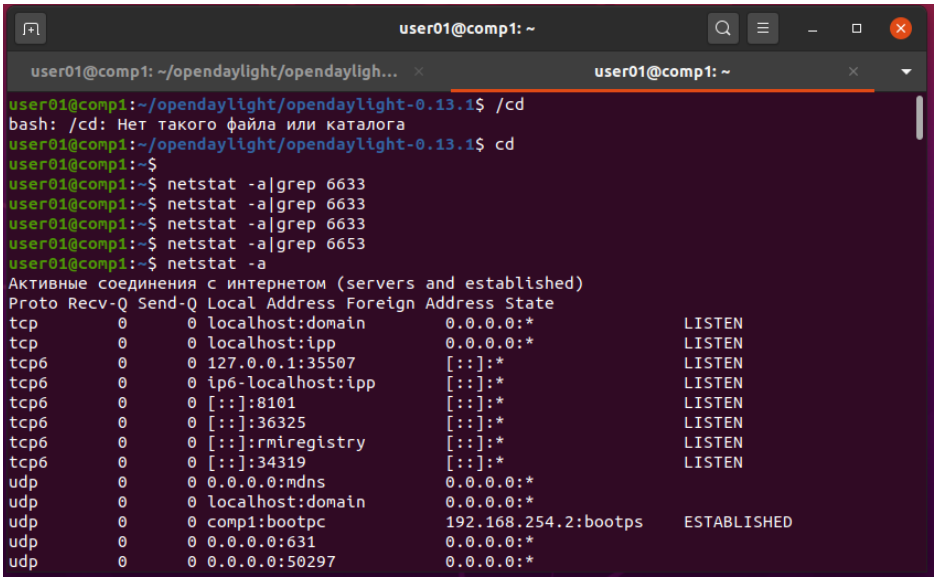

Рис 3.11. Порти які слухає контролер

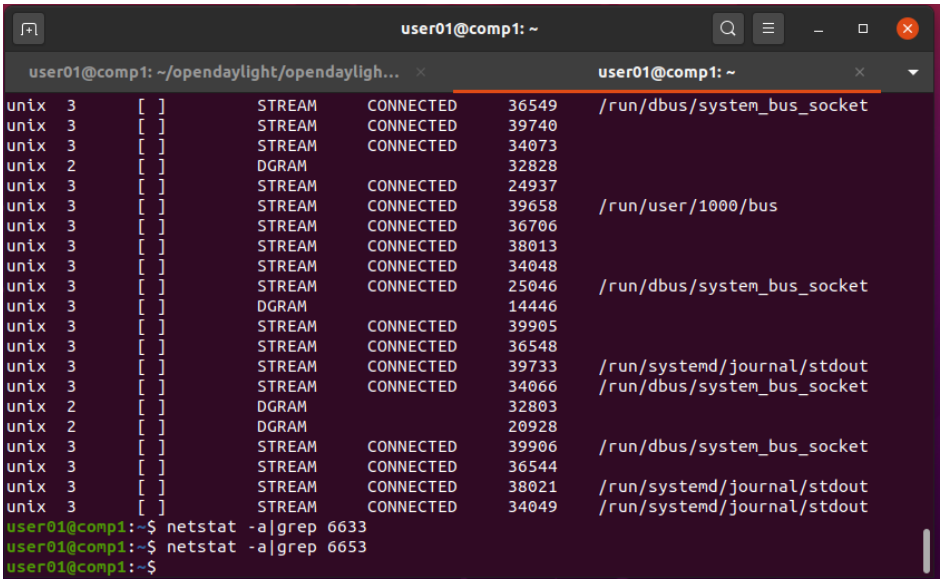

Рис 3.12. Порти які слухає контролер

Як бачимо контролер не має такий портів для прослуховування.

```
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
( ip link del s1-eth1;ip link del s1-eth2 ) 2> /dev/null
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
mininet@mininet-vm:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:5c:d7:cc
          inet addr:192.168.56.3  Bcast:192.168.56.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7030 (7.0 KB)  TX bytes:5328 (5.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:535 errors:0 dropped:0 overruns:0 frame:0
          TX packets:535 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:41872 (41.8 KB)  TX bytes:41872 (41.8 KB)

mininet@mininet-vm:~$
```

Рис 3.13.Мережеві налаштування Mininet

```
user01@comp1: ~
user01@comp1:~$ sudo apt install net-tools
[sudo] пароль для user01:
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Уже установлен пакет net-tools самой новой версии (1.60+git20180626.aebd88e-1ubu
ntu1).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакето
в, и 253 пакетов не обновлено.
user01@comp1:~$
```

Рис 3.14 Команда встановлення мережного пакету

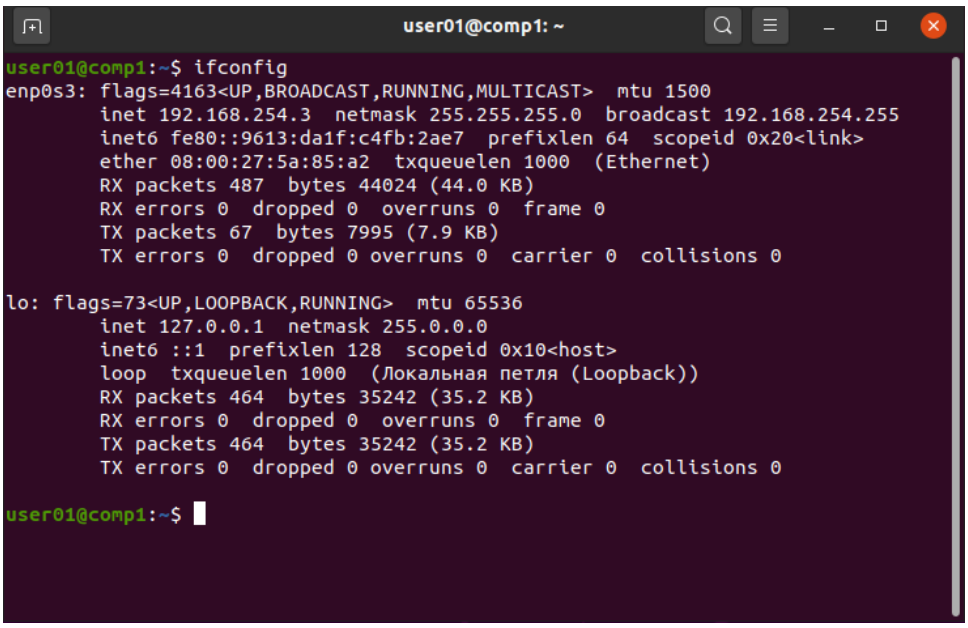

Рис 3.15.Мережеві налаштування контролеру

Висновки

1. Зроблений огляд можливостей Oracle VM VirtualBox, приведено встановлення Mininet на VM, описані основні функції Mininet такі як відображення параметрів запуску, для роботи з хостами та комутаторами, для запуску простого веб-сервера та клієнта. Створення кастомної топології.
2. Встановлення Ubuntu, та встановлення контролеру OpenDayLight.
3. Інтеграція контролеру OpenDayLight в Mininet.

РОЗДІЛ 4
РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Удосконаленяя методу обробки потоків в системі SDN та розгортання мережі SDN з зовнішнім контролером може мати місце в розробкі додатків в системі SDN. Данний спосіб не потребує великих обчислювальних ресурсів та може бути застосована в додатках різного масштабу.

Таблиця 5.1

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Удосконалений метод обробки потоків в системі SDN, розгортання мережі SDN з зовнішнім контролером	Розгортання мережі SDN	Можливість швидко розгортувати мережу SDN для тестування додатків на контролері
	Відмова від використання сторонніх сервісів шляхом розгортання власної платформи	Можливість розгорнути власний сервіс на базі шаблону та відмовитись від щомісячної підписки або раптових сбоїв в роботі сервісу

5.2 Можливості запуску проекту

Таблиця 5.2

Попередня характеристика потенційного ринку стартап-проекту

	Показники стану ринку	Характеристика
	Кількість головних конкурентів	>30
	Динаміка ринку	Ринок додатків з керуванням потоків постійно зростає.
	Наявність та характер обмежень для входу	Велика кількість конкурентів може буди компенсована рішенням використовувати метод пріоритезації потоків
	Специфічні вимоги до стандартизації та сертифікації	Для використання додатку має використовуватися зовнішній контролер який підтримує OpenFlow1.3

Таблиця 5.3

Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в швидкодії роботи програмно-керованої мережі	Самостійні та корпоративні клієнти	Корпоративні клієнти вимагають оптових тарифів, або розгортання власного сервісу	Ідеальне співвідношення ціна-якість

Таблиця 5.4

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Зависока конкуренція	Багато різних рішень провокують постійні зміни в політиці конкурентів	Необхідність слідкування за конкурентами та проводження регулярніх опитувань користувачів
2	Обмежений функціонал	Націленність на конкретну задачу може бути недоліком	Сертифікувати усіма можливими засобами, щоб зробити продукт більш надійним порівняно з конкурентами

5.3 Технологічний аудит

Таблиця 5.5

Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технології
1	Удосконалений метод обробки потоків в системі SDN, розгортання мережі SDN з зовнішнім контролером	Mininet	Є в наявності	Доступні безкоштовно
2		Контролер OpenDayLight	Є в наявності	Доступні безкоштовно
3		Ubuntu 20.4	Є в наявності	Доступні безкоштовно

5.4 Розроблення ринкової стратегії продукту

Таблиця 5.6

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті
1	Приватні компанії	Не всі готові прийняти	Є в наявності	Висока
2	Державні установи	Не готові прийняти	Є в наявності	Низька
3	Навчальні та дослідницькі програми	Не всі готові прийняти	Є в наявності	Низька

Таблиця 5.7

Визначення базової стратегії розвитку

	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкуренто-спроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
	Демонстрація можливостей продукту шляхом публікації науково-популярних статей та відеоматеріалів в соцмережах	Співпраця з цільовими групами	Вільне розповсюдження для некомерційних користувачів	Оновлення продукту відповідно до побажань клієнтів та відповідно до оновлень конкурентів

Таблиця 5.8

Визначення базової стратегії розвитку

№	Чи є проект першим в своєму роді на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде продукт копіювати основні характеристики конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Пошук нових користувачів та заохочення існуючих	Основний функціонал буде схожим	Відстеження нових технологій та адаптація продукту

5.5 Розроблення маркетингової стратегії стартап-проекту

Таблиця 5.9

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Часте оновлення програми	Врахування побажань клієнтів та виправлення помилок в найкоротші терміни	Створення відділу підтримки для аналізування реагування на відгуки клієнтів
2	Формування звітів з результатами моделювання	Можливість формувати звіти по роботі програми з детальним описом процесу моделювання та оцінкою ефективності мережі, що тестувалася	Модуль для формування звітів з результатами роботи програми

Таблиця 5.10

Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="2">Удосконалений метод обробки потоків в системі SDN, розгортання мережі SDN з зовнішнім контролером</td></tr>
<tr><td rowspan="6">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Розмір</td></tr>
<tr><td>1. Модуль впровадження додатку в контролер</td><td>5МБ</td></tr>
<tr><td>2. Модуль графічного інтерфейсу присутній в зовнішньому контролері</td><td>20МБ</td></tr>
<tr><td>3. Модуль аналізу результатів та статистики за допомогою мережевих утиліт</td><td>5МБ</td></tr>
<tr><td colspan="2">Якість: внутрішнє тестування програми, логування збоїв та система зворотнього зв’язку для повідомлення про неналежну роботу програми</td></tr>
<tr><td colspan="2">Пакування: продаж електронних ключів-ліцензій та інструкцією, що надсилається на електронну адресу замовника</td></tr>
<tr><td colspan="2">Марка: назва організації-розробника + назва товару</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="2">До продажу: реалізація додатку</td></tr>
<tr><td colspan="2">Після продажу: електронна версія додатку з ключем</td></tr>
<tr><td colspan="3">За рахунок чого потенційний товар буде захищено від копіювання: прив’язка копії програмного продукту до конкретного ПК та активація програми шляхом введення ліцензійованого ключа, або шляхом надання послуг без передачі програмного продукту замовнику</td></tr>
</table>

Таблиця 5.11

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі становлення ціни на товар/послугу
1	0$ - 100$	0-100	>100000$/місяць	20-100

Таблиця 5.12

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Покупка ліцензії на продукт або договір про надання послуг без передачі ПО до замовника	Розміщення на власному сайті для електронної дистрибуції	Канал збуту однорівневий (через роздрібну дистрибуцію)	Вертикальна (право власності залишається у розробника ПЗ)

Таблиця 4.13 Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Купують товар на вимогу	Тематичні зустрічі, конференції, презентації тощо	Надання послуг тестування мереж розробленим продуктом, продаж ПЗ	Демонстрація основних функцій розробленого продукту	Охоплення аудиторії, пояснення функцій та можливостей розробленого продукту та його переваг

Висновки

П’ятий розділ присвячений розробці стартап-проекту для розроблюваного продукту, зроблено опис ідеї проекту, проведено детальний аналіз конкурентів та потенційних можливостей для реалізації продукту на даному ринку. Для дослідження в рамках розробки стартап-проекту було виконано наступні завдання:

1. Наведено загальний опис ідеї проекту для виходу на ринок, описаний приблизний функціонал проекту, визначено конкурентів та проведено порівняння функціоналу продуктів від конкурентів для визначення переваг та недоліків розроблюваного продукту в порівнянні з вищезгаданими конкурентами.
2. Проведено технічний аудит проекту і визначено можливості реалізації даного проекту.
3. Проаналізовано ринкові можливості для запуску проекту, наведено порівняльну характеристику перспектив запуску з конкурентами, створено план та ринкову стратегію розвитку продукту, визначено цільові групи та способи просування проекту в маси.
4. Розроблено маркетингову програму для просування продукту на ринку, описано способи та основні напрямки збуту, визначно першочергові цільові групи та маркетингові стратегії для розширення клієнтської бази.

В підсумку, було отримано стартап-проект для запуску розробленого програмного продукту на ринок, отримано навички створення стартап-проектів, побудови маркетингової стратегії та аналізу обраного ринку.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Проведено огляд технології програмно-конфігурованих мереж, розглянута архітектура SDN, яка складається з трьох рівнів: рівень інфраструктури, рівень управління, рівень додатків. Рівень інфраструктури включає в себе набір передавальних мережевих пристроїв, які доступні через уніфікований інтерфейс і виконують інструкції таблиць потоків для передачі мережевого трафіку. Рівень управління регулює обмін інформацією про таблиці маршрутизації і виробляє правила маршрутизації на основі закладених в контролер алгоритмів аналізу потоків пакетів, що пересилаються від комутаторів. Рівень додатків складається з високорівневих додатків, які взаємодіють з мережевим контролером або набором контролерів.

2. Описаний відкритий стандарт протокол OpenFlow та мережевий пристрій, який підтримую роботу по протоколу OpenFlow – комутатор OpenFlow. Відкритий стандарт описує вимоги, що пред'являються до комутатора, що підтримує протокол OpenFlow для віддаленого управління За допомогою сучасних маршрутизаторів зазвичай вирішуються дві основні задачі: передача даних (forwarding) - просування пакета від вхідного порту на певний вихідний порт і управління даними - обробка пакету і прийняття рішення про те, куди його передавати далі, на основі поточного стану маршрутизатора.

3. Описані компоненти OpenFlow комутатора такі, як канал OpenFlow,два підхода побудови каналу Out-of-band та In-band, OpenFlow порт - мережевий інтерфейс для передачі пакетів між пристроями з підтримкою OpenFlow. Описаний формат записів в таблиці мережевих протоколів : match, actions, priority, timeouts, метадані. Процес обробки на основі таблиці, конвеєрна обробка пакета.

Розглянутий віртуальний комутатор Open vSwitch - найбільш універсальне рішення, підтримує найбільш популярні Гіпервізор, включаючи KVM, Virtual Box, Xen та XenServer. Open vSwitch складається зі служби-

комутатора (user-space) і супроводжуючого модуля ядра (kernelspace), який керує процесом потокової (flowbased) комутації. Open vSwitch підтримує широкий набір технологій, включаючи NetFlow, sFlow, port mirroring, VLAN, LACP, TLS / SSL. Описана структура комутатору, яку умовно можна розділити на дві частини – user-space і kernel-space.

4. Приведені основні функції та вимоги до контролера OpenFlow.

До основних функцій відносяться : керування ресурсами сервера, забезпечення зв'язку між рівнем додатків і мережею, надання сервісів, керування комутацією, створення додатків на основі API. До основних вимог до контролера відноситься гнучкість масштабованість тимчасова розширюваність продуктивність.

5. Описані основні модулі контролера SDN: модуль виявлення посилань, модуль топології, модуль зберігання даних, модуль створення стратегій, модуль таблиці потоків та модуль керуючих даних та особливості контролера SDN такі, як мова програмування, підтримка OpenFlow, мережева програмованість, ефективність, southbound interfaces, northbound interfaces, партнерство.

6. Розглянуті найбільш популярні контролери такі, як OpenDayLight описана архітектура, POX контролер описана POX API, Floodlight контролер описана структура.

7. Приведені підсумки та призначення цих контролерів та прийнято рішення використовувати OpenDayLight. POX контролер використовується для перевірки ефективості ПКС на різних топологіях мереж і тестування протоколу OpenFlow. Floodlight і OpenDaylight можна об'єднати в одну категорію, як два проекти, що розробляються на мові Java. Хоча Floodlight можна назвати перехідним між POX і OpenDaylight, тому що останній підтримує будь-які топології мереж, велика кількість протоколів маршрутизації, є більш гнучким і краще підходить для вирішення реальних бізнес-завдань.

8. Описані способи тестування додатків контролера, серед вимоги до віртуальних мереж: гнучкими, здатними підтримувати різні топології, архітектури маршрутизації і пере адресації, а також незалежні конфігурації; керованими ,щоб відокремлювати політику, яку намагається вказати мережевий; масштабованими для збільшення кількості співіснують віртуальних, мереж; захищеними шляхом виділення різних логічних мереж друг від друга; гетерогенними, для підтримування різних технологій.

9. Розглянуто середовище для моделювання Mininet, описаний функціонал та можливості емулятору : надає простий і недорогий мережевий випробувальний стенд для розробки додатків OpenFlow, дозволяє одночасно, окремо, працювати декільком розробникам над однією і тією ж топологією, забезпечує комплексне тестування топології без необхідності підключення до фізичних мереж, включає інтерфейс командного рядка з підтримкою топології і OpenFlow для налагодження або запуску загальномережевих тестів, підтримує довільні кастомні топології і включає в себе основний набір параметрезованих топологій, та переваги перед системами засновані на повній віртуалізації системи, і апаратними тестовими системами.

10. Зроблений огляд можливостей Oracle VM VirtualBox такі як переносимість, не потрібна апаратна віртуалізація, гостьові доповнення: загальні папки, безшовні вікна, 3D віртуалізація, багатопоколінні розгалужені знімки, VM групи. Приведено встановлення Mininet на VM, описані основні функції Mininet такі як відображення параметрів запуску, для роботи з хостами та комутаторами, для запуску простого веб-сервера та клієнта. Створення кастомної топології.

11. Підготовка операційної системи Ubuntu, та встановлення контролеру OpenDayLight на VM.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Open Networking Foundation. SDN architecture Issue 1 June, 2014 ONF TR502.

https://www.opennetworking.org/images/stories/downloads/sdnresources/technical- reports/TR_SDN_ARCH_1.0_06062014.pdf,

2. Open Networking Foundation. SDN Architecture Overview Version 1.0 December 12, 2013. https://www.opennetworking.org/images/stories/downloads/sdnresources/technical- reports/SDN-architecture-overview-1.0.pdf,

3. Open Networking Foundation. OpenFlow Switch Specification Version

1.5.1 (Protocol version 0x06) March 26, 2015 ONF TS-025. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow- switchv1.5.1.pdf,

4. Программно-конфигурируемые сети - № 09, 2012 «Открытые системы» Издательство «Открытые системы» URL: http://www.osp.ru/os/2012/09/13032491/

5. “OpenFlow switch specification: Version 1.4.0”. https://www.opennetworking.org/images/stories/downloads/specification/openflow- spec-v1.4.0.pdf,

6. Software-Defined Networking: The New Norm for Networks ONF White Paper April 13, 2012. https://www.opennetworking.org/images/stories/downloads/sdnresources/white-papers/wp-sdn-newnorm.pdf,

7. Шалимов А.В. Технологии SDN/OpenFlow http://lvk.cs.msu.su/~sveta/SDN_OpenFlow_basics_lecture1.pdfSDN&NFV,

8. Смелянский Р.Л Технологии SDN и NFV: новые возможности для телекоммуникаций. https://www.osp.ru/netcat_files/userfiles/Ethernet_2015/Smelyansky_R.pdf,

9. F. Alencar, M. Santos, M. Santana and S. Fernandes, '"How Software Aging affects SDN: A view on the controllers," *Global Information Infrastructure and Networking Symposium (GIIS), 2014*, pp. 1-6.

10. Feng Wang, Heyu Wang, Baohua Lei and Wenting Ma, '"A Research on High-Performance SDN Controller," Cloud Computing and Big Data (CCBD), 2014 International Conference on, pp. 168-174.

11. O.N. Fundation, '"Software-defined networking: The new norm for networks," *ONF White Paper*.

12. Ashton, Metzler. "Ten Things to Look for in an SDN Controller." *white paper [online]*. Available at: www. necam. com/Docs (2013).

13. Sridhar Rao, "SDN Series Part Eight: Comparison of Open Source SDN Controllers" [online]. Available at: http://thenewstack.io/sdn-series-part-eight-comparison-of- open-source-sdn-controllers/.

14. "How Do SDN Southbound APIs Work?" [Online]. Available at: https://www.sdxcentral.com/resources/sdn/southbound- interface-api/.

15. "What are SDN Northbound APIs?" [Online]. Available at: https://www.sdxcentral.com/resources/sdn/north-bound- interfaces-api/.

16. M. Jammal, T. Singh, A. Shami, R. Asal and Y. Li, '"Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, no. 0, 10/29, pp. 74-98.

17. Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, D. Niyato and Haiyong Xie, '"A Survey on Software-Defined Networking," *Communications Surveys & Tutorials, IEEE*, vol. 17, no. 1, pp. 27-51.

18. B.A.A. Nunes, M. Mendonca, Xuan-Nam Nguyen, K. Obraczka and T. Turletti, '"A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617-1634.

19. D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C. Esteve

Rothenberg, S. Azodolmolky and S. Uhlig, "'Software- Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76.

20. Fei Hu, Qi Hao and Ke Bao, "'A Survey on Software- Defined Network and OpenFlow: From Concept to Implementation," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 2181-2206.

21. H. Farhady, H. Lee and A. Nakao, "'Software-Defined Networking: A survey," *Computer Networks*, vol. 81, no. 0, 4/22, pp. 79-95.

22. Документация Opendaylight.
https://www.opendaylight.org/,

23. Документация VirtualBox.
https://www.virtualbox.org/manual/

24. OpenvSwitch URL: https://sdnblog.ru/what-is-open-vswitch/

25. Foundations of Modern Networking SDN, NFV, Qoe, IoT, and Clound / William Stallings — Addison-Wesley Professional; 1 edition (November 8, 2015)

26. SDN and NFV Simplified: A Visual Guide to Understanding Software Defined Networks and Network Function Virtualization / Jim Doherty —Addison-Wesley Professional; 1 edition ,March 12, 2016

27. Software-Defined Networking (SDN) with OpenStack / Sriram Subramanian, Sreenivas Voruganti — Packt Publishing 2016

28. "OpenFlow switch specification: Version 1.4.0". https://www.opennetworking.org/images/stories/downloads/specification/openflow- spec-v1.4.0.pdf,

29. R. Sherwood and et al., "Flowvisor: A network virtualization layer,"
OpenFlow Switch Consortium, Tech. Rep, 2009

30. A. Khurshid et al., "VeriFlow: verifying network-wide invariants in real time," in Proc. of HotSDN, 2012.

http://www.tik.ee.ethz.ch/file/9ee69e89be779fd1448a7356d79ddb18/openflow_sec. pdf,

31.OpenFlow: A Security Analysis Rowan Kloti ¨ ETH Zurich Zurich, Vasileios Kotronis ETH Zurich Zurich, Paul Smith AIT Austrian Institute of Technology 2444 Seibersdorf

32. Open vSwitch [Electronic resourse]. – Mode of access: openvswitch.org/ (Accessed 1 June 2016).

33. .Sherwood R., Yap K.-K. Cbench (controller benchmarker) [Electronic resourse] / R.Sherwood, K.- K.Yap. – Mode of access: http://www.openflowswitch.org/wk/index.php/Oflops.

34. .Sherwood R., Yiakoumis Y. Oftrace: An OpenFlow Debugging and Analysis Tool [Electronic resourse] / R.Sherwood, Y.Yiakoumis // Stanford Clean Slate Lab, 2009. – Mode of access: archive.openflow.org/wk/images/f/fb/Oftrace.pdf (viewed on June 10, 2016).

35. Mininet: An Instant Virtual Network on your Laptop (or other PC) – Mininet [Electronic resourse]. – Mode of access:mininet.org/ (Accessed 1 June 2016).

36. Стрихалюк Б.М. Алгоритми пошуку шляху за критерієм мінімальної затримки для центру обробки даних / Б.М.Стрихалюк, О.М.Шпур, М.О.Селюченко, Т.В.Андрусів // Вісник Національного університету «Львівська політехніка» “Радіоелектроніка та телекомунікації”. – Львів, 2014. – №796. – С.176-181 (Index Copernicus, Google scholar).

37. Klymash M.A. Novel approach of optimum multicriteria vertical handoff algorithm for heterogeneous wireless networks / M.Klymash, B.Stryhalyuk, I.Demydov, M.Beshley, M.Seliuchenko // International Journal of Engineering and Innovative Technology (IJEIT). – 2014. – Vol.4. – Issue 5(4). – P.42-52 (Index Copernicus).